读者注意：本文档内容仅限于说明1.486版本的设计与实现，易达保留在后续版本中更改设计和实现的权利。从1.486版本开始，易达将逐步支持现货交易，本文档对现货相关功能的说明均会用“现货”加以指明，如未特别指明，则表明该内容仅适用于期货交易所。

1 快速入门

本章将通过一个简单的策略程序，向投资者演示接入并使用易达柜台的必要步骤，帮助投资者建立易达API的基本概念。

1.1 环境准备

首先从https://www.hanlinit.com/download下载对应版本的API。下载前请先注册官网投资者账号，待通过审批后即可下载（易达承诺注册信息仅用于发布与易达有关信息，不用于其他用途）。下载完成后解压ydClient_1_486_XX_XX.tgz文件，演示使用ydClient/win64和ydAPI目录中的工具和程序。然后通过易达的Windows GUI客户端ydClient直接连接柜台，通过ydClient能方便检查后续编写的程序的运行结果。

首先进入ydClient/win64目录，将YDConfig.ini修改为以下内容：

```
##########################################
######### Network configurations  #########
##########################################

# IP address of yd trading server
# TCP port of yd trading server, other ports will be delivered after logged in

TradingServerIP=118.190.175.212
TradingServerPort=41600

##########################################
######### Trading configurations  #########
##########################################

# Choose trading protocol, optional values are TCP, UDP, XTCP
TradingProtocol=TCP

# Timeout of select() when receiving order/trade notifications, in millisec. -1 indicates running without select()
TradingServerTimeout=10

# Affinity CPU ID for thread to send TCP trading and receive order/trade notifications, -1 indicates no need to set CPU affinity
TCPTradingCPUID=-1

# Affinity CPU ID for thread to send XTCP trading, -1 indicates no need to set CPU affinity
XTCPTradingCPUID=-1

##########################################
####### MarketData configurations  ########
##########################################

# Whether need to connect to TCP market data server
ConnectTCPMarketData=yes

# Timeout of select() when receiving TCP market data, in millisec. -1 indicates running without select()
TCPMarketDataTimeout=10

# Affinity CPU ID for thread to receive TCP market data, -1 indicate no need to set CPU affinity
```

```
TCPMarketDataCPUID=-1

# Whether need to receive UDP multicast market data
ReceiveUDPMarketData=no

##########################################
####### Misc configurations        ########
##########################################

AppID=yd_dev_1.0
AuthCode=ecbf8f06469eba63956b79705d95a603
```

上述配置信息指向一套易达提供的上期所、能源所互联网测试环境。该环境7*24小时运行，会播放某一特定交易日的生产行情，方便投资者调试程序，详情请参见https://www.hanlinit.com/docs/dev-environments/。

配置完成后，双击运行YDClient.exe，并使用001、002-099、100中的任一账户（口令同用户名）登录，登录成功后可以浏览各个菜单中的内容。

完成上述步骤后，下面将开始准备策略程序。

1.2 策略程序

要编译可执行程序，需要先准备好易达样例程序。请将ydAPI/linux64、ydAPI/linclude以及ydAPI/example下的所有文件都拷贝到Linux服务器的工作文件夹下面，并使用下列命令编译：

```
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c ydExample.cpp -o ydExample.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example1.cpp -o Example1.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example2.cpp -o Example2.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example3.cpp -o Example3.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example4.cpp -o Example4.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example5.cpp -o Example5.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example6.cpp -o Example6.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example7.cpp -o Example7.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example8.cpp -o Example8.o
g++ -fpic -g -std=c++11 -c -O3 -pthread -Wall -c Example9.cpp -o Example9.o
g++ -g -std=c++11 -o ydExample -I. ./ydExample.o ./Example1.o ./Example2.o ./Example3.o
./Example4.o ./Example5.o ./Example6.o ./Example7.o ./Example8.o ./Example9.o -m64 -Wall -
lpthread -lrt -ldl -Wl,-rpath,. -L. -l:yd.so
```

编译成功后，将上述拷贝到YDConfig.ini的内容同样拷贝到config.txt文件中，然后就可以运行第一个样例程序了，其中username和password使用登录ydClient的用户名和口令替换：

```
# ./ydExample <example name> <config file> <username> <password> <instrumentID>
./ydExample Example1 config.txt <username> <password> cu2306
```

当出现以下信息时，即表示策略程序开始执行，其中的告警信息请参考信息采集中关于dmidecode的内容：

```
sudo: a password is required
login successfully
Position=0
sell open 1 at 71580
```

发生交易后，请在ydClient的账户明细界面中观察报单、成交、持仓和资金的变化情况。

至此，第一个易达样例程序已经编译并运行成功，投资者可以继续尝试其他样例程序并阅读这些程序，有助于帮助投资者快速掌握API的基础知识。关于ydExample各个例子的说明摘抄如下：

```
所有样例程序都使用下面这个针对单一品种的策略：
```

```
    如果行情中的买量比卖量大100就按照对手价买入1手
    如果行情中的卖量比买量大100就按照对手价卖出1手
    风控限制是持仓量不允许超过3

以下样例程序给出不同精细程度的持仓管理方式：
Example1:只管理成交持仓，不管理报单持仓，同时只允许至多一张挂单
Example2:基于报单回报管理报单持仓，进行更加精确的持仓管理
Example3:在上个例子的基础上，又增加了基于自身发出的报单来管理报单持仓，以便实现更加精确的持仓管理。此外，该例子也展示了利用notifyCaughtUp获取追上最新流水的信息
Example4:使用YDExtendedApi来实现相同的功能，程序可以大大简化
Example9:展现如何使用OrderGroup的方式来实现可靠的UDP报单，并使用本地号撤单

以下样例程序用于展现其他功能：
Example5:展现发送期权执行、期权放弃执行和询价指令
Example6:展现一个做市商报价系统如何使用YDExtendedApi
Example7:展现如何使用YDExtendedApi进行简单的风险监控和报警
Example8:展现如何使用YDExtendedApi进行自动建立传统组合持仓
```

通过学习样例程序，投资者应该已经掌握了编写易达策略程序的基本步骤。接下来，建议投资者系统性的阅读文档剩下的内容，文档不仅仅详细说明了易达的基本原理和使用方法，还提供了编写易达API程序的技巧和最佳实践。

如在阅读过程中遇到任何问题，欢迎投资者通过邮件（support@hanlinit.com）或者经纪商与易达支持团队取得联系，易达支持团队很乐意回答投资者的任何问题。

2 基本概念

2.1 API选型

易达提供了ydApi、ydExtendedApi、裸协议、ydCTP以及python接口共五种不同类型的交易接口，以满足广大投资者的丰富需求。

裸协议是自由度最高的报单方式，易达向全市场公布了报单和撤单上行协议以及报单回报、成交回报、错单回报等下行协议，投资者可以自行构建UDP报文报撤单或者监听解析下行回报报文，使用时需要配合ydApi或者ydExtendedApi。适合习惯使用裸协议接入柜台的投资者，详情请参考裸协议。

ydCTP是仿CTP的API，模拟了CTP的报单、查询等核心部分功能，使得原先基于CTP开发的程序不经修改即可连接易达柜台，便于尚未正式使用易达的投资者评估、测试易达交易系统的性能和稳定性。由于易达和CTP的体系结构并不兼容，ydCTP无法覆盖CTP的所有接口，也无法做到和原生接口行为完全一致，不建议在生产环境中正式使用ydCTP交易。详情请参考相关文档。

易达python版本API请参考客户端包中的相关文档。

2.1.1 ydApi

ydApi是高性能API，以极简高效为设计理念。负责把投资者的报单送到柜台以及接收到回报后通过回调函数通知投资者，通常情况下API内部不会留存除了静态数据以外的报单、成交等动态数据，更不会帮助投资者计算资金和持仓，因此内存占用极小。使用ydApi的投资者需要自行根据日初持仓以及后续的报单、成交等回报来管理资金和持仓以及提供相应的查询方法。适合高度关注性能且有自行管钱管仓的能力的投资者。

若开启了高可用特性，ydApi会占用与回报数据量大致相当的内存用于高可用恢复。经估测，在Linux64位系统中500万报单无成交的情况下ydApi约占400M内存，实际生产中由于存在成交、错单、组合等，在相同报单量的情况下会占用更多。

ydApi拥有以下特性：

- 线程安全：即在任意线程可以随时调用任何API方法
- 可重入：即在任意回调函数中可以调用任何API方法
- 指针稳定：即ydApi中通过get或者find方法获得的静态数据的指针在整个生命周期内都不会改变，策略程序可以随时读取指针指向的数据；但通过notify回调接口返回的报单回报、成交回报等指针是不同的，OrderSysID相同的多次回报的指针也是不同的。特例是通过notifyCombPosition返回的静态数据指针是稳定的。

易达目前提供Linux64位、Win32位和Win64位三种版本的API。Linux64位版本的API针对Linux平台做了大量优化，其性能大大优于Windows的两个版本，是易达推荐在生产中使用的版本。对于必须要使用Windows平台的投资者，由于Win32位程序只能使用2G的用户内存，为防止超出内存限制导致API卡死的情况发生，建议投资者使用64位版本。

2.1.2 ydExtendedApi

ydExtendedApi是全能型API，以功能完善和使用方便为设计理念。在继承了ydApi所有功能和特性的基础上（ydExtendedApi是ydApi的子类），API内部保存了所有报单、成交等动态数据，基于这些动态数据帮助投资者管理资金和持仓，并提供了各种丰富的特有数据结构和查询方法，代价是微小的下行性能开销和一定空间占用，当交易量大时占用的空间较大。适合习惯使用全功能API的投资者。

由于ydExtendedApi在本地维护了扩展数据结构，占用的空间相比ydApi较大。经估测，在Linux64位系统中500万报单无成交的情况下，未启用高可用特性时ydExtendedApi约占1400M内存；启动高可用特性时ydExtendedApi约占1800M内存，实际生产中由于存在成交、错单、组合等，在相同报单量的情况下会占用更多。

ydExtendedApi的查询方法都在本地执行，不会到柜台查询，但是由于所有查询都会加锁而导致成本较高，策略程序可以保存从API返回的指针，并在后续执行过程中直接使用指针指向的数据。除了继承了ydApi的指针稳定特性之外，凡是通过ydExtendedApi的get或者find方法获取的，以及YDExtendedListener回调返回的扩展动态数据的指针也都是稳定的。

2.2 报单方式

易达支持TCP、UDP和XTCP三种报单方式，这些报单方式在穿透性能和可靠性方面有所区别，投资者可以根据自身需求选择报单方式。

2.2.1 TCP报单

TCP报单整体穿透延时比较高，API报单自身的耗时（调用API报单接口到报单第一个字节被发送到光纤上的耗时）约为1-2微秒左右，柜台穿透在8-10微秒左右，但能保证报单必定送达柜台。投资者在测试或者通过互联网连接到柜台时，建议采用TCP的方式，可以避免因为防火墙等原因导致的发单丢失问题。

在TCP报单相同的线程内，还负责发送非交易信息（如修改口令）和接收柜台发送的回报信息，因此无论采用何种报单方式，TCP报单所在线程一定会被创建。

要使用TCP报单，请在API配置文件中设置TradingProtocol=TCP。详情请参见配置文件。

2.2.2 UDP报单

UDP报单整体穿透延时低，API报单自身的耗时约为200纳秒左右，柜台穿透约2-3微秒左右（视不同交易所穿透有所区别），易达官方的参考穿透值是在UDP报单方式下测得的，建议投资者在生产环境中使用UDP报单。UDP报单仅用于报送上行报单，回报仍然通过原TCP通道返回。

对于部分投资者担心UDP可能会丢单的问题，首先，易达柜台所在的网络条件通常是稳定、快速和空闲的，除非模块、光纤等发生故障，否则极少会发生UDP丢失的情况；其次，易达提供了柜台回报功能，投资者收到柜台回报即代表柜台已经收到了报单，这样即便丢失了投资者也能知道，详情请参见报单回报中关于柜台回报的功能描述。

要使用UDP报单，请在API配置文件中设置TradingProtocol=UDP。详情请参见配置文件。若柜台没有开启UDP服务，那么每次调用报单接口都将返回false。UDP没有专门的发单线程，将直接在调用报撤单接口的线程中直接发单，因此请将调用线程绑定在隔离核上以确保发单性能。

2.2.3 XTCP报单

XTCP报单整体穿透延时较低，API报单自身的耗时约为250纳秒左右，柜台穿透相比UDP报单增加100ns以内的延时。其运行模式与UDP相同，仅用于报送上行报单，回报仍然通过原TCP通道返回。

相比UDP报单，XTCP完全消除了丢单的问题且增加穿透时间较小，适合于不信任UDP协议的投资者使用。但由于TCP连接协议栈维护的开销相比UDP较大，因此建议投资者谨慎使用XTCP连接，若大范围使用XTCP线程，将会对策略机和柜台造成较大压力。

要使用XTCP报单，请在API配置文件中设置TradingProtocol=XTCP。详情请参见配置文件。若柜台没有开启XTCP服务，那么将会降级使用TCP报单。由于XTCP服务默认是关闭的，请在使用前与经纪商确认是否打开了XTCP服务。绝大多数情况下，XTCP将在调用报撤单接口的线程中直接发单，因此请将调用线程绑定在隔离核上以确保发单性能；TCP重发报文以及TCP心跳报文通过XTCP的专属线程发出，可以通过XTCPTradingCPUID参数绑定。

从`1.486`版本开始，XTCP支持主从模式（master-slave）的多网卡绑定（network bonding）。只要在操作系统上配置了主从模式的绑定网卡，且参与绑定的网卡都是易达支持的高性能网卡，例如Solarflare X2522，Exanic X10，API可以自动识别并支持绑定网卡的高可用切换。

2.3 API线程

易达API启动完成后，会创建TCP回报接收、TCP行情接收和定时器三个线程；报单是通过调用insertOrder的线程直接发出的，并没有专门的线程处理报单发送。

2.3.1 TCP回报接收线程

TCP回报接收线程的主要工作是接收回报并回调YDListener以及向柜台发送心跳，YDListener或者YDExtendedListener的回调基本（notifyMarketData除外）都是从本线程发起，若长时间（60秒）停留在回调函数中不退出会引起心跳超时而导致本线程的TCP连接断线，因此建议投资者在每个回调函数的处理时间尽可能短。可以在API配置文件中设置TCPTradingCPUID指定需要绑定CPU核心。

TCP回报接收线程有忙查询和select两种网络监听模式。若采用忙查询模式，则收取回报的速度较快，但是其所在CPU的使用率会达到100%；若采用select模式，则会根据设定时间等待select超时后继续下一次select，对CPU几乎没有开销，但是收取回报的速度较忙查询慢，可以在API配置文件中设置TradingServerTimieout设置超时时间，具体请参考交易配置。

为保证柜台上所有连接接收流水的公平性，柜台每向一个连接推送20条流水，就会切换到下一个连接，这样就可以避免某账户盘中登录后收取全量流水时对其他已有连接接收回报产生影响。

2.3.2 TCP行情接收线程

TCP行情接收线程的主要工作是接收交易所前置推送的行情并回调YDListener以及向柜台发送心跳，YDListener中的notifyMarketData的回调是从本线程发起的，与TCP回报线程相同的原因，notifyMarketData的处理也需要快速，否则会导致本线程的TCP连接断线。可以在API配置文件中设置TCPMarketDataCPUID指定需要绑定CPU核心。

TCP行情接收线程有忙查询和select两种网络监听模式。若采用忙查询模式，则收取行情的速度较快，但是其所在CPU的使用率会达到100%；若采用select模式，则会根据设定时间等待select超时后继续下一次select，对CPU几乎没有开销，但是收取行情的速度较忙查询慢几微秒，可以在API配置文件中设置TCPMarketDataTimeout设置超时时间，具体请参考行情配置。

2.3.3 定时器线程

定时器线程是统一执行API内各个定时任务的线程，该线程每隔1毫秒定时唤醒。每次定时器触发都将询问各个定时任务是否执行，由各个定时任务自身的特性决定是否执行。目前定时器线程仅用于资金刷新机制中自动模式通知投资者合适的刷新时机。

定时器线程不可关闭，可以在API配置文件中设置TimerCPUID绑定该线程到指定CPU。

2.4 用户自定义字段

易达API支持本地和远程两种类型的用户自定义字段。本地用户自定义字段只会保存在API本地内存中，不会发送到柜台；远程用户自定义字段会通过报单、报价发送到柜台，并在报单回报、报价回报和成交中带回。

2.4.1 本地用户自定义字段

为方便投资者将用户数据存储在易达结构体中，基于易达指针稳定的特性，易达在YDExchange、YDProduct、YDInstrument、YDMarketData、YDCombPositionDef、YDAccount、YDAccountExchangeInfo、YDAccountProductInfo、YDAccountInstrumentInfo中提供了pUser、UserFloat、UserInt1和UserInt2四个字段。

API**永远**不会修改设置在这些字段中的数据，包括以下特殊场景：

- 调用startDestory()销毁API时，不会主动释放存储在pUser的空间
- 当发生主备切换时，由于上述结构体并不会有变化，因此存储在用户自定义字段中的数据也不会被改变

若上述字段不足以存储用户数据，可定义结构体并将该结构体的指针存储在pUser中。本质上，API提供了一片长度24字节的连续自定义空间，在这片空间内可以存放任何值，即可以打破API预定义的字段边界跨字段存储数据。投资者可以修改头文件中的字段名称、类型和个数以符合实际需要，但需确保总长度不超过24字节。

2.4.2 远程用户自定义字段

从1.486版本开始，报单和报价中增加了一个32字节的UserRef字段，该UserRef会通过报单回报、报价回报和成交回报带回。API和柜台均不会使用和修改UserRef的值。

虽然OrderRef和UserRef都是供投资者填写并可从各种回报中带回的远程用户自定义字段，但不同的是OrderRef存放是顺序性信息，会参与报单、报价递增性检查，因此，并不适合通过编码等方式在OrderRef中存放分类性信息。UserRef适合存放分类性、标识性信息，例如策略号、服务器编号等信息，32字节的字段长度为分段存储多个信息提供了较为充足的空间。

2.5 系统保留字段

在API的各结构体中，存在较多系统保留字段，字段名中含有SystemUse的字段即为系统保留字段。

系统保留字段是API用于内部处理逻辑的字段，投资者不应该读取或修改系统保留字段中的值。易达会随着版本升级改变系统保留字段的用法，其中存储值的含义也会发生改变。修改系统保留字段会导致不可预知的严重后果，易达不对投资者擅自修改系统保留字段中的值产生的任何后果负责。

2.6 版本规则

易达采用四段式版本号编码规则，格式为a.b.c.d，每个发布版本只会修改四段版本号中的一个，易达承诺如下：

- a：协议版本号，改变本段版本号的版本必定会修改协议并导致之前发布的API版本全部不兼容，也可以包含b/c段的修改内容，只有当API和柜台的本段版本号一致且API的a.b.c版本号低于柜台的a.b.c版本号时，API才能在该版本的柜台上正常工作。本段版本号较大的版本是高版本，若相同则继续比较b段。
- b：功能版本号，增加本段版本号的版本必定会增加新功能，也可以包含c段的修改内容。本段版本号较大的版本是高版本，若相同则继续比较c段。
- c：补丁版本号，增加本段版本号的版本只能包含修复Bug的补丁内容，且必须尚未发布的版本。本段版本号较大的版本是高版本，若相同则继续比较d段。
- d：紧急补丁版本号，增加本段版本号的版本**原则上**只能包含修复紧急Bug的补丁内容，且必须是已正式发布的生产版本，但是为了应对快速变化的业务需求，易达可能会不得不在紧急补丁中增加新的功能，但是会保证版本兼容性。本段版本号较大的版本是高版本。

易达承诺所有投资者使用的易达的版本是相同的，不存在所谓的特殊版本。但是按照易达的升级策略和规范，在试运行和升级过程中，部分投资者可能会使用到尚未向全市场正式发布的高版本柜台，请各位投资者谅解：

- 在完成版本内部测试后，易达会邀请个别愿意并有能力承担风险，且在业务和技术方面均拥有较强运维和应急处置能力的经纪商试运行新版本，试运行时间并不固定，版本发布的内容越多试运行时间通常会越长，而且一旦在试运行阶段修复了新的问题或引入了新的功能，试运行时间会进一步延长
- 在试运行通过后，易达会向全市场发布正式版本，此时所有经纪商可以分批次逐步升级到最新版本。为了保证易达有充足的服务人力应对升级过程中的各种问题，在开始的1-2周会限制每周的总升级套数。以易达柜台目前的市场保有量以及部分经纪商采取了较为保守的升级策略，全市场完成升级需要2-3个月时间。

2.6.1 API版本兼容性

只有当API和柜台的a段版本号相同，且API的a.b.c段版本号介于MinApiVersion和MaxApiVersion之间，API是才能正常连接和登录柜台，否则会收到YD_ERROR_TooHighApiVersion=62或者YD_ERROR_TooLowApiVersion=58的登录报错信息。

- 若经纪商未做特别设定，柜台默认的MaxApiVersion等于柜台自身的版本。在特殊情况下，可以通过配置柜台参数MaxApiVersion修改柜台支持的最高API版本，MaxApiVersion只能设置a.b.c段版本号。只比较a.b.c段版本号是为了能让API的生产紧急补丁（修改版本号第四位）适配低版本的柜台。
- 若经纪商未做特别设定，柜台默认的MinApiVersion为1.0.0。

柜台当前版本、MaxApiVersion和MinApiVersion的获取方式参见系统参数。

API的版本可以通过以下两种不同的方法获取，也可以在log日志中查看API版本号，详情参见写日志：

```
class YDApi
{
  virtual const char *getVersion(void)
}

/// Same as getVersion inside YDApi, put here to get version without make api
YD_API_EXPORT const char *getYDVersion(void);
```

3 生命周期

3.1 创建

要创建API实例，首先需要确定使用ydApi还是ydExtendedApi，这两种API的概要差异可以参考API选型，详细差异请查阅本文相关内容。

创建API的进程不允许调用任何会创建子进程的系统调用，例如fork()、system()、exec()、vfork()、clone()、posi_spawn()等，以免产生无法预期的问题。

3.1.1 创建ydApi

创建ydApi有以下两种方法，makeYDApi的参数需要填入配置文件路径，API会从路径指向的文件中读取配置内容；makeYDApiFromConfig的参数则直接填入配置文件的内容本身，内容格式应与前一接口中指向的配置文件格式相同，即参数名和参数值组成的配置对，并用`\n`分隔。

```
YDApi *makeYDApi(const char *configFilename)
YDApi *makeYDApiFromConfig(const char *configDesc)
```

若选择使用ydApi，那么创建过程大致如下：

```
// 创建YDApi
YDApi *pApi=makeYDApi(configFilename);
if (pApi==NULL)
{
    printf("can not create API\n");
    exit(1);
}

// 创建Api的监听器
YDExampleListener *pListener=new YDExampleListener(...);
/// 启动Api
if (!pApi->start(pListener))
{
    printf("can not start API\n");
    exit(1);
}
```

makeYDApi需要传入一个配置文件，该配置文件的配置方法和样例参见配置文件。

上面例子中的YDExampleListener是投资者自行编写的YDListener的子类，所有回报类信息都通过该类的回调函数通知到策略程序，由于YDListener的实例是由策略程序创建的，策略程序应该维护其生命周期，在需要的时候自行销毁。YDListener中的回调函数将分布在本文各个功能章节中分别介绍。

调用ydApi.start即可启动API，参数pListener不可以为空，该函数调用后会立即返回并不阻塞。为防止程序直接退出，策略程序需要调用成功后增加阻塞代码。

```
virtual bool start(YDListener *pListener)
```

3.1.2 创建ydExtendedApi

创建ydExtendedApi有以下两种方法，makeYDExtendedApi的参数需要填入配置文件路径，API会从路径指向的文件中读取配置内容；makeYDExtendedApiFromConfig的参数则直接填入配置文件的内容本身，内容格式应与前一接口中指向的配置文件格式相同，即参数名和参数值组成的配置对，并用`\n`分隔。

```
YDExtendedApi *makeYDExtendedApi(const char *configFilename)
YDExtendedApi *makeYDExtendedApiFromConfig(const char *configDesc)
```

若选择使用ydExtendedApi，那么创建过程大致如下：

```
// 创建YDApi
YDExtendedApi *pApi=makeYDExtendedApi(configFilename);
if (pApi==NULL)
{
    printf("can not create API\n");
    exit(1);
}
// 创建Api的监听器
YDExampleListener *pListener=new YDExampleListener(...);
/// 启动Api
if (!pApi->start(pListener))
{
    printf("can not start API\n");
    exit(1);
}
```

可以看到，除了调用makeYDExtendedApi创建API实例外，其余部分与ydApi完全相同。

为方便ydExtendedApi的用户获得扩展消息的变动通知，在启动API时可以使用startExtended同时接收YDListener和YDExtendedListener的回调通知。

```
// YDExample7Listener同时实现了YDListener和YDExtendedListener接口
class YDExample7Listener: public YDListener, public YDExtendedListener {}

// 创建YDApi
YDExtendedApi *pApi=makeYDExtendedApi(configFilename);
if (pApi==NULL)
{
    printf("can not create API\n");
    exit(1);
}
// 创建Api的监听器
YDExample7Listener *pListener=new YDExample7Listener(...);
/// 启动Api，这里使用了YDExtendedListener
if (!pApi->startExtended(pListener,pListener))
{
    printf("can not start API\n");
    exit(1);
}
```

startExtended的函数签名如下，参数pListener和pExtendedListener均不可以为空，除了增加了YDExtendedListener回调通知之外，本函数与start没有任何区别：

```
virtual bool startExtended(YDListener *pListener,YDExtendedListener *pExtendedListener)
```

YDExtendedListener的定义如下，与YDListener返回的结构体相比，YDExtendedListener返回的扩展结构体包含了更多信息，可以方便投资者编写代码。各个扩展结构体的具体内容请参见ydDataStruct.h。

```
class YDExtendedListener
{
public:
    virtual ~YDExtendedListener(void)
```

```
    {
    }
    // all address of parameters in following methods are fixed
    virtual void notifyExtendedOrder(const YDExtendedOrder *pOrder)
    {
    }
    virtual void notifyExtendedTrade(const YDExtendedTrade *pTrade)
    {
    }
    virtual void notifyExtendedQuote(const YDExtendedQuote *pQuote)
    {
    }
    virtual void notifyExtendedPosition(const YDExtendedPosition *pPosition)
    {
    }
    virtual void notifyExtendedAccount(const YDExtendedAccount *pAccount)
    {
    }
    // notifyExchangeCombPositionDetail and notifyExtendedSpotPosition will only be used when trading SSE/SZSE
    virtual void notifyExchangeCombPositionDetail(const YDExtendedCombPositionDetail *pCombPositionDetail)
    {
    }
    virtual void notifyExtendedSpotPosition(const YDExtendedSpotPosition *pSpotPosition)
    {
    }
};
```

3.1.3 配置文件

在调用makeYDApi方法时需要指定客户端使用的配置文件。配置文件中的参数主要包括三类：

- 网络配置，包括ydServer的IP地址和端口，其中端口请始终填写经纪商提供的TCP端口，其他端口包括UDP报单端口、TCP行情端口会在登录后自动下发，是否启用UDP报单是由UDPTrading参数控制的。
- 交易配置，包括是否使用UDP报单，以及接收回报线程的工作方式。
- 行情配置，包括是否接收ydServer的TCP或者UDP行情。

以下为ydApi提供的用于生产环境的客户端配置文件最佳实践模板，该模板最终实现的效果是：

- 使用UDP报单
- 忙查询接收TCP回报，加快回报的获取速度，并把接收线程绑定在3号CPU
- 打开了资金重算RecalcMode功能，ConnectTCPMarketData=no的设置会被覆盖为yes，即接收行情

```
##########################################
######### Network configurations  #########
##########################################

# Count of recovery site. Used to achieve high availablity at the expense of a little performance of order notification.
# 0 for no recovery, 1 for recovery always use primary site, 2 for recovery use primary and secondary sites
RecoverySiteCount=0

# IP address of primary trading server
TradingServerIP=127.0.0.1

# TCP port of primary trading server, other ports will be delivered after logged in
```

```
TradingServerPort=51000

# IP address of secondary trading server.
# Valid only when RecoverySiteCount equals to 2.
TradingServerIP2=

# TCP port of secondary trading server, other ports will be delivered after logged in.
# Valid only when RecoverySiteCount equals to 2.
TradingServerPort2=

##########################################
######### Trading configurations  #########
##########################################

# Choose trading protocol, optional values are TCP, UDP, XTCP
TradingProtocol=UDP

# Affinity CPU ID for thread to receive order/trade notifications, -1 indicate no need to set CPU affinity
TCPTradingCPUID=-1

# Affinity CPU ID for thread to send XTCP trading, -1 indicate no need to set CPU affinity
XTCPTradingCPUID=-1

# Timeout of select() when receiving order/trade notifications, in millisec. -1 indicates running without select()
TradingServerTimeout=-1

# Work mode for recalculation of margin and position profit. Valid when using ydExtendedApi.
#     auto(default): subscribe market data and automatically recalculate in proper time.
#     subscribeOnly: subscribe market data and recalcMarginAndPositionProfit should be called explicitly
#     off: never do recalculation
RecalcMode=auto

# Gap between recalculations, in milliseconds. Valid when RecalcMode is set to auto.
# It will be adjusted to 1000 if less than 1000
RecalcMarginPositionProfitGap=1000

# Delay of recalculation after market data arrives to avoid collision with input order, in milliseconds.
# Valid when RecalcMode is set to auto. Should be between 0 and 100.
RecalcFreeGap=100

##########################################
####### MarketData configurations  ########
##########################################

# Whether need to connect to TCP market data server
ConnectTCPMarketData=no

# Timeout of select() when receiving TCP market data, in millisec. -1 indicates running without select()
TCPMarketDataTimeout=10

# Affinity CPU ID for thread to receive TCP market data, -1 indicate no need to set CPU affinity
TCPMarketDataCPUID=-1

```

```
# Whether need to receive UDP multicast market data
ReceiveUDPMarketData=no

##########################################
######### Other configurations  ###########
##########################################

AppID=yd_dev_1.0
AuthCode=ecbf8f06469eba63956b79705d95a603
```

3.1.3.1 网络配置

网络配置包含易达柜台的IP地址和端口，生产环境交易时，请向所在经纪商技术部门询问具体配置情况。

RecoverySiteCount：高可用节点个数。当为0时，不启用API高可用特性；当为1时，启用单柜台高可用特性，当柜台重启后，API会重新连接柜台并尽可能恢复到最新状态，从而不会导致API直接退出；当为2时，启用主备双柜台高可用特性，当主柜台因故障停止备用服务器（在TradingServerIP2和TradingServerPort2中指定）启动后，API会重新连接柜台并恢复到最新状态。使用高可用特性，会损失极为微小的回报性能。

TradingServerIP：易达柜台的IP地址

TradingServerPort：易达柜台的端口。易达柜台通常会开放三个端口，分别是TCP交易、TCP行情和UDP交易，XTCP交易和UDP行情端口默认关闭。TradingServerPort需要填入的是TCP交易的端口，其他端口会在登录成功后由柜台自动下发到客户端。请不要将UDP交易端口填入TradingServerPort，否则会导致无法连接易达柜台，如果你希望通过UDP或者XTCP报单，请通过TradingProtocol=UDP或者TradingProtocol=XTCP参数控制。

UDPTradingServerPort：UDP交易的端口。由柜台自动下发，通常不要填写。当需要从外网访问内网柜台而做了NAT时，对外的UDP交易端口可能会发生改变，此时下发的端口将无法正确接收UDP报单数据，需要根据经纪商提供的实际端口配置该参数。

XTCPTradingServerPort：XTCP的交易端口。由柜台自动下发，通常不要填写。当需要从外网访问内网柜台而做了NAT时，对外的XTCP交易端口可能会发生改变，此时下发的端口将无法正确接收XTCP报单数据，需要根据经纪商提供的实际端口配置该参数。

TCPMarketDataServerPort：TCP行情的端口。由柜台自动下发，通常不要填写。当需要从外网访问内网柜台而做了NAT时，对外的行情端口可能会发生改变，此时下发的端口将无法正确接收行情，需要根据经纪商提供的实际端口配置该参数。

TradingServerIP2：易达备用服务器的IP地址。RecoverySiteCount为2时生效。

TradingServerPort2：易达备用服务器的端口。设置方法同TradingServerPort。RecoverySiteCount为2时生效。

UDPTradingServerPort2：易达备用服务器的UDP交易端口，使用方法同UDPTradingServerPort。RecoverySiteCount为2时生效。

XTCPTradingServerPort2：易达备用服务器的XTCP的交易端口，使用方法同XTCPTradingServerPort。RecoverySiteCount为2时生效。

TCPMarketDataServerPort2：易达备用服务器的TCP行情端口，使用方法同TCPMarketDataServerPort。RecoverySiteCount为2时生效。

3.1.3.2 交易配置

TradingProtocol：报单方式，可选报单方式为TCP、UDP、XTCP。由于UDP和XTCP总体穿透性能相比TCP好很多，建议在生产环境使用UDP或XTCP报单。为保证兼容性，若没有设置TradingProtocol，则会使用原有配置UDPTrading。

UDPTradingType：UDP交易的发送器实现方式，可选值为automatic、sf、exa和socket，不指定该参数时默认为automatic。socket表示使用标准的socket协议栈，sf表示使用solaflare的ef_vi，exa表示使用exanic，automatic表示自动选择合适的发送器实现方式，可以是sf、exa、socket中的一种。由于生产环境较为复杂，存在其他应用程序和易达API的发送器实现不兼容的情况，会导致API自动降级为socket发送，从而大大降低API发送性能。因此，建议使用UDP交易的投资者强制指定对应的发送器实现方式，一旦遇到冲突会有显著的错误提示或者易达API应用无法启动，使得问题更容易被发现，避免投资者在不知情情况下性能大幅降级的问题。

TCPTradingCPUID：设置TCP发送报单和接收回报线程的亲和性。-1代表不需要设置亲和性；否则为CPU/Core的编号。

XTCPTradingCPUID：设置XTCP发送报单线程的亲和性。-1代表不需要设置亲和性；否则为CPU/Core的编号。

TradingServerTimeout：设置该参数指示YDListener的TCP交易接收线程如何使用监听网络通信。如果设置为正整数值，表示该线程使用POSIX的select机制去监听是否有网络数据到达，监听时间间隔为所指定的值，单位为毫秒。设置为-1，接收线程将以非阻塞方式读取信息，其CPU使用率会达到100%，如果客户端未根据自身机器的配置情况合理分配CPU/Core，设置线程与CPU/Core的亲和性，将严重影响客户端程序的运行性能，延迟会显著增加。

RecalcMode：保证金和持仓盈亏的重算模式。默认值为auto。

- auto：完全由API负责重算保证金和持仓盈亏，API会自动订阅计算需要的行情。可以通过RecalcMarginPositionProfitGap和RecalcFreeGap这两个参数控制刷新间隔和距离行情的延时时间。会强制接收TCP行情
- subscribeOnly：只帮助自动订阅需要的行情，但是需要由投资者自行调用recalcMarginAndPositionProfit重算方法。会强制接收TCP行情
- off：不重算保证金和持仓盈亏。通常投资者不应该使用这个模式

RecalcMarginPositionProfitGap：两次重算之间的间隔，单位为毫秒，默认为1000毫秒。最小值为1000，如果小于1000，系统自动会调整为1000。本参数只有当RecalcMode为auto时才有效。

RecalcFreeGap：距离行情的延时间隔。为了尽可能避免与报单发生冲突，要尽可能避免在行情到来的时候重算，投资者可以根据不同交易所调整参数使得行情在两个时间切片报单较为空闲的时间重算。单位为毫秒，默认值为100毫秒，必须介于0和100之间，否则大于100的值会被调整为100，小于0的值会被调整为0。本参数只有当RecalcMode为auto时才有效。

OperWay：指定现货交易中的操作方式，如果输入的操作方式不在该投资者允许的操作方式范围内，则不允许登录，只有在现货系统中有效。大部分证券公司只支持单字符的操作方式，个别证券公司需要用到双字符或更多的操作方式，按照证券公司要求填写即可。

3.1.3.3 行情配置

请注意，无论是TCP行情还是UDP组播行情，易达的行情均不是高速行情，目前仅可用于易达服务端和客户端用于计算保证金和盈亏。如在生产交易中需要行情，请联系经纪商接收组播行情。

ConnectTCPMarketData：是否连接ydServer的TCP行情服务。yes表示接收，no表示不接收。

TCPMarketDataTimeout：设置该参数指示YDListener的TCP行情接收线程如何使用监听网络通信。如果设置为正整数值，表示该线程使用POSIX的select机制去监听是否有网络数据到达，监听时间间隔为所指定的值，单位为毫秒。设置为-1，接收线程将以非阻塞方式读取信息，其CPU使用率会达到100%，如果客户端未根据自身机器的配置情况合理分配CPU/Core，设置线程与CPU/Core的亲和性，将严重影响客户端程序的运行性能，延迟会显著增加。

TCPMarketDataCPUID：设置TCP行情接收线程的亲和性。-1代表不需要设置亲和性；否则为CPU/Core的编号。

ReceiveUDPMarketData：是否接收ydServer发送的UDP组播行情。目前所有易达柜台的UDP组播行情功能均是关闭的，请始终保持该参数为no。

3.1.3.4 其他配置

TimerCPUID：用于设置API中定时器线程的CPUID。

AppID和AuthCode：投资者可以在配置文件中设置AppID和AuthCode，并在登录调用login时appID和authCode分别填入NULL，API就会使用配置文件中配置的值。

LogDir：指定ydApi产生日志的目录，默认为log。目录不会自动产生，需要手工创建，如果没有对应的目录存在，则不会产生日志文件。

3.1.3.5 自定义配置

易达支持读取配置文件中的配置内容，投资者即可以读取易达API的配置信息，投资者又可以将自定义的配置放在易达配置文件中，然后通过易达API读取。

当投资者设定自定义配置项时，请以"应用名.参数名"的格式予以命名，例如MyApp.Username，如果直接输入没有应用名的参数，该参数仍然是可以使用的，但是API在启动时会报告该参数没有被使用的警告信息，为了避免误解，不建议使用没有应用名分隔的参数。

易达自定义配置支持列表类型的参数，可以设置多个同名参数实现这个效果。设置样例如下所示：

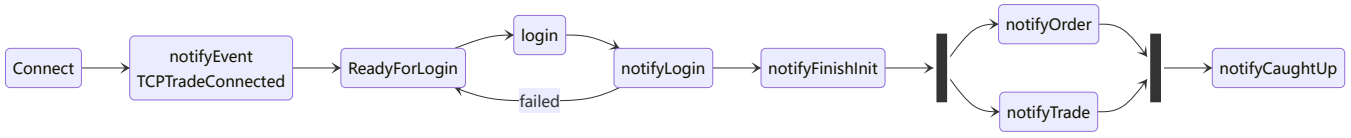
```
MyApp.TradeExchange=SHFE
MyApp.TradeExchange=INE
MyApp.TradeExchange=CFFEX
MyApp.TradeExchange=DCE
MyApp.TradeExchange=CZCE
```

易达API提供了以下两个API获取自定义参数的方法。第一个方法可以获取单个自定义参数，当该参数是列表参数时，获取的是该列表的第一个设置值。第二个方法可以获取列表参数的所有参数值，返回的结果集在用完后需要投资者手动调用destroy()销毁，当用于单个自定义参数时将返回包含单个元素的YDQueryResult集合，用完后同样需要调用destroy()销毁由API分配的空间。

```
/// get first config using this name in config file (parameter of makeYDApi or makeYDExtendedApi), NULL if not found
virtual const char *getConfig(const char *name)
/// get all configs using this name,  user should call destroy method of return object to free memory after using
virtual YDQueryResult<char> *getConfigs(const char *name)
```

3.2 启动

在创建了API实例并调用start或者startExtend启动API后，API会按照以下启动流程完成启动初始化工作。



3.2.1 连接

实例启动后，就会持续连接TradingServerIP和TradingServerPort指定的柜台地址和TCP端口，直到连接成功后，会通过notifyEvent(YD_AE_TCPTradeConnected)回调消息通知已连接成功。

若API配置了双柜台高可用模式，那么会轮询TradingServerIP和TradingServerPort指定的主柜台地址和TCP端口以及TradingServerIP2和TradingServerPort2指定的备柜台地址和TCP端口，直到连接成功后，会通过notifyEvent(YD_AE_TCPTradeConnected)回调消息通知已连接成功。

3.2.2 登录

连接成功后，会立即通过回调函数notifyReadyForLogin通知策略程序API已准备好，可以尝试登录。该回调函数的参数hasLoginFailed是指发起本次notifyReadyForLogin回调的原因是否为上次登录失败，若原因是上次登录失败，用户没有必要用完全相同的信息再次登录，因为结果必定失败。

```
virtual void notifyReadyForLogin(bool hasLoginFailed)
```

策略程序在收到notifyReadyForLogin回调通知后，通常应该立即调用login发起登录。login函数的签名如下。

```
virtual bool login(const char *username,const char *password,const char *appID,const char *authCode)
```

username应填写资金账号，password应填写资金账号对应的口令，关于口令更多的信息请参见口令。

期货公司受到保证金监控中心的监管，因此在期货公司部署的柜台系统中，需要输入合法的appID和authCode，否则无法成功登录。期货公司的appID和authCode的详细填写说明请参见穿透式监管。

证券公司并不是保证金监控中心的监管对象，因此在证券公司部署的柜台系统中，appID和authCode可以设置为NULL，柜台不会校验这两个参数。为满足证券公司个性化的需求，appID可以用来上报个性化信息：

- 国君证券：国君证券的投资者可以输入终端信息中的扩展信息，易达将自动改写转义符，并把最后一个下划线替换为分号。例如投资者输入YDClient_V1.0，易达将在投资者终端信息的扩展信息中记为YDClient;V1.0

下面是常见的登录代码样例，appID和authCode定义在配置文件中，因此下例中上述两个参数填NULL：

```
virtual void notifyReadyForLogin(bool hasLoginFailed)
{
    if (!m_pApi->login(m_username,m_password,NULL,NULL))
    {
        printf("can not login\n");
        exit(1);
    }
}
```

登录结果会通过notifyLogin回调返回：

```
virtual void notifyLogin(int errorNo,int maxOrderRef,bool isMonitor)
```

若notifyLogin的errorNo为0，则表示登录成功。isMonitor为true则表示为当前登录用户为柜台管理员，普通投资者账户登录则必然为false。maxOrderRef表示在登录时刻柜台0号报单组OrderGroupID已收到的最大OrderRef，如需获取所有报单组的最大OrderRef请参考报单组。除了以下三种情况外，无论报单、报价单成功与否，只要新报单、报价单所在报单组的OrderRef大于该组当前的MaxOrderRef，都将修改当前账户的MaxOrderRef为该报单的OrderRef：

- 报单、报价单因为报单号单调递增检查而导致的错单不会计入MaxOrderRef
- 报单、报价单因柜台接收到的网络数据包错误而导致无法识别报单账户的错单不会计入MaxOrderRef，通常不应发生本错误，多见于投资者错误破解我司协议后的报单、报价单
- 在不支持报价指令的柜台（没有做市商功能的授权）上报入报价单而导致的错单不会计入MaxOrderRef

登录成功时，API就能收到柜台下发的TCP行情端口和UDP交易端口，因此，易达不需要在配置文件中指定行情端口、UDP交易端口与XTCP交易端口。但是在某些场景下，如从互联网访问柜台时，通常经过映射后的互联网端口与从柜台下发的端口是不同的，如果不做处理，会表现为TCP行情连不上或者UDP交易无法发送到柜台。在这种情况下，可以手工设置TCP行情端口或者UDP交易端口以覆盖自动下发的端口，可使用TCPMarketDataServerPort覆盖行情端口，UDPTradingServerPort覆盖UDP交易端口，XTCPTradingServerPort覆盖XTCP交易端口；若配置了双柜台高可用模式，可以使用TCPMarketDataServerPort2覆盖备用服务器的行情端口，UDPTradingServerPort2覆盖备用服务器的UDP交易端口，XTCPTradingServerPort2覆盖备用服务器的XTCP交易端口。

若notifyLogin的errorNo非0，则表示登录失败。errorNo会告知具体失败的原因，此时maxOrderRef和isMonitor都是无意义的，请不要使用。API会等待3秒钟后继续回调notifyReadyForLogin等待用户发送登录请求，此时notifyReadyForLogin中的hasLoginFailed会被设置为true。登录失败的原因可能是以下几种：

错误码	错误定义	说明
12	YD_ERROR_InvalidClientAPP	AppID或AppAuthCode错误

错误码	错误定义	说明
18	YD_ERROR_AlreadyLogined	当前API已完成登录，不可以重复登录
19	YD_ERROR_PasswordError	登录口令错误
20	YD_ERROR_TooManyRequests	登录请求数量超过限制
21	YD_ERROR_InvalidUsername	登录用户不存在
27	YD_ERROR_InvalidAddress	客户端IP地址不在允许登录的地址段，仅对管理员登录生效
35	YD_ERROR_ClientReportError	穿透式监管信息采集失败。详情请参见穿透式监管。
50	YD_ERROR_TooManyLogines	易达柜台的授权限制了同时可以登录柜台的账户数，当达到登录上限后，后续连接的账户将无法登录，必须要等待现有账户退出登录后，其他账户才能继续登录。一个账户无论有多少个连接，都只计算一个登录账户数，换而言之，只有当账户所有的连接都退出后，即YDAccount.LoginCount为0，该账户才能被视为退出。
58	YD_ERROR_TooLowApiVersion	API版本低于柜台要求的最低版本，详情参见API版本兼容性
62	YD_ERROR_TooHighApiVersion	API版本高于柜台支持的最高版本，详情参见API版本兼容性
99	YD_ERROR_PasswordNotSet	登录口令没有设置

3.2.2.1 口令

易达柜台支持本地口令认证和统一认证两种方式，期货公司和多数证券公司一般使用柜台本地保存的口令认证，而有些证券公司建立了统一认证系统并要求柜台接入并使用该系统认证投资者登录时输入的口令。

易达柜台自行管理的本地口令可以设置复杂度的有效期，而当柜台使用统一认证时，口令复杂度和有效期设置无效，不可以使用API的修改口令方法。下面关于口令复杂度和口令有效期的说明仅限于本地口令。

3.2.2.1.1 口令复杂度

易达口令复杂度包括最小口令长度和口令字符集数量两种：

- 口令最小长度指修改口令时新口令必须达到的最小长度，但是由于易达柜台的最大口令长度为64字节，因此口令长度应位于最小口令长度和64字节之间
- 口令字符集数量指修改口令时口令中必须包含的字符集类型数量。字符集类型包括数字、小写字母、大写字母、符号四种，其中符号只能是（+-*/'`"@#_$%&|~^ ()[]{}\,.:;!?<>=）。字符集数量限制是0-4之间的数字，0表示无限制，1-4表示口令中必须要有n种字符集。若设置为0无限制，则可以全部使用四种字符集外的字符作为口令

投资者和管理员的最小口令长度和口令字符集数量均可独立设置，设置值从系统参数中下发。

在修改口令时柜台会检查新口令是否符合复杂度要求，若通不过则会收到YD_ERROR_WeakPassword=111的错误。管理员重置投资者口令时，不会被检查是否符合复杂度要求，因此可以设置任何口令。

3.2.2.1.2 口令有效期

默认情况下，投资者和管理员的口令是永久有效的。但是经纪商可以分别设置投资者和管理员的口令有效期限，当口令到期后，投资者仍然可以登陆柜台，但是此时只能修改口令，而无法使用任何交易指令。口令修改成功后，可以使用当前API实例直接开始交易，无需重新连接和登录。

易达柜台通过以下回调方法在口令到期后通知投资者和管理员修改口令，以及提醒投资者和管理员口令剩余有效期。该方法在登陆成功时被调用。

```
virtual void notifyServerNotice(const YDServerNotice *pServerNotice)
```

当口令过期时，柜台会向API推送口令已到期的消息，上述回调方法中的pServerNoticed的ServerNoticeType被设置为YD_SNT_PasswordExpired，此时投资者或者管理员必须修改口令才能继续交易。

当口令有效期小于15天时，柜台会向API推送口令到期日临近的消息，上述回调方法中的pServerNoticed的ServerNoticeType被设置为YD_SNT_PasswordNearExpiration，PasswordNearExpirationInfo.RemainDayCount被设置为剩余天数。

上述两类消息会被同时写入API的日志中。

3.2.2.2 穿透式监管

根据监管要求，穿透式监管主要分为两个部分，一是登录时需要指定客户端系统的AppID和AuthCode，二是需要上报登录系统所在服务器的信息。下面将分别介绍这两个机制。

3.2.2.2.1 AppID和AuthCode

AppID和AuthCode是策略系统和中继系统等使用API接入柜台的客户端系统的用户名和口令，客户端系统要连入柜台首先得满足经纪商的接入测试要求，测试通过后经纪商会将该系统的AppID和AuthCode加入到柜台中，此后投资者才能使用该客户端系统连入柜台交易。中继系统应使用自身的AppID，而不是中继客户端的AppID。

> 易达提供的ydClient客户端有其固化在客户端内部的AppID和AuthCode，该客户端的AppID和AuthCode也被默认添加在柜台的允许系统清单中，因此任何投资者都可以使用ydClient登录到柜台。

易达并不限制AppID的命名格式，但监管对AppID有要求，建议按照监管要求命名客户端系统。具体要求如下：

- AppID由三部分组成，分别是厂商名称、软件名和版本号，各部分最大长度分别为10、10、8个字符
- AppID应遵循以下格式：厂商名称_软件名称_版本号，例如yd_client_1.0
- 对于个人开发的终端软件，厂商名称应固定为client。由于使用易达系统的大部分投资者都是自研系统，因此厂商名称应固定为client

对于期货公司部署的柜台系统，login方法中的appID和authCode有两种设置方式：

- 若appID和authCode设置为NULL，API会使用配置文件中指定的AppID和AuthCode，如果配置文件中没有配置这两个参数则会出错
- 若appID和authCode设置为具体的值，则API只会使用接口中指定的参数而不会使用配置文件中指定的值

默认情况下，客户端系统的AppID与投资者账户并没有绑定关系，即客户端系统可以被所有投资者共享使用。但实际上大多数接入易达柜台交易的客户端系统是由投资者自行开发的专属策略系统，应经纪商的合规性要求，自`1.486`版本起易达柜台提供了资金账号和AppID绑定的功能，经纪商为AppID配置了绑定资金账号后，只有被绑定的资金账号才能使用该AppID。若登陆时使用了非建立绑定关系的AppID，会收到YD_ERROR_InvalidClientAPP的错误。

3.2.2.2.2 信息采集

监管需要采集的信息包括两部分，一部分是必须从客户端服务器上采集，一部分可以在柜台端采集。易达API会自动从客户服务器上采集监管要求的客户端服务器信息，不同类型的终端操作系统采集的信息有所区别，下面仅列出从客户端服务器上采集的信息：

```
Windows:终端类型(固定为1)@信息采集时间@私网IP1@私网IP2@网卡MAC地址1@网卡MAC地址2@设备名@操作系统版本@硬盘序列号@CPU序列号@BIOS序列号@系统盘分区信息
Linux:终端类型(固定为2)@信息采集时间@私网IP1@私网IP2@网卡MAC地址1@网卡MAC地址2@设备名@操作系统版本@硬盘序列号@CPU序列号@BIOS序列号
```

默认情况下，API会在动态链接库被加载的时候收集穿透式监管信息，可以通过增加环境变量YD_DCINFO将穿透式监管信息采集的时间延后到API启动时，YD_DCINFO可以是任意值。

> 在Linux系统的采集过程中，易达API会使用dmidecode程序采集BIOS序列号，通常若不做特殊设置普通用户是没有该程序的执行权限的，运行客户端系统的用户必须具有该程序的执行权限才能确保信息采集完全。易达会首先检查dmidecode是否被赋予了suid，若被赋予则直接执行，若没有被赋予，则使用sudo dmidecode的方式执行。投资者可以通过chmod +x /usr/sbin/dmidecode命令为该程序添加suid。

当柜台参数"客户端报告检查方式ClientReportCheck"被设置为强制enforce时，若柜台检查采集信息发现采集不完整时，例如上面第二条Linux样例中的硬盘序列号未采集到，会禁止客户端登录，并产生YD_ERROR_ClientReportError的登录错误。下面列出了易达API采集信息采用的操作系统命令，若发现采集信息有缺失的，请手工执行以下对应的采集方法并确认命令执行结果是否正常。请使用与启动客户端系统相同操作系统用户执行命令，确保能发现权限不足导致的问题（大部分取不到数据的原因）：

- Windows的硬盘序列号：Get-WmiObject -Query "SELECT SerialNumber FROM Win32_PhysicalMedia"
- Windows的CPU序列号：Get-WmiObject -Query "SELECT ProcessorID FROM Win32_Processor"
- Windows的BIOS序列号：Get-WmiObject -Query "SELECT SerialNumber FROM Win32_BIOS"
- Linux的硬盘序列号：首先通过/bin/lsblk -dn -o TYPE,NAME找到TYPE为disk的设备名NAME，然后调用/sbin/udevadm info –query=all –name=/dev/{NAME}获得该设备的序列号
- Linux的BIOS序列号：/usr/sbin/dmidecode -s system-serial-number

3.2.2.3 交易日

交易日会随着登录回报一起传回到API，从notifyLogin回调开始投资者即可以使用getTradingDay获取交易日。

```
virtual int getTradingDay(void)
```

易达柜台的交易日是根据系统时间计算得出的（日盘时设置为当前自然日的日期，夜盘24:00:00前为下一非节假日的日期，夜盘24:00:00后，若当前自然日为交易日，则为当前自然日，若当前自然日为节假日，则为下一非节假日的日期），生产环境计算得到的交易日通常是准确的，但是在测试环境中很有可能与YDMarketData.TradingDay是不一致的，导致不一致的原因可能是使用了过去的日初数据、交易所的测试环境没有切换交易日或者在周末测试的时候人为制造的交易日。

3.2.3 收取静态数据

为减少柜台查询对柜台产生的冲击，易达所有的查询都是在本地执行的，这样就要求客户端和服务端有完全相同的数据，这一原则对ydApi和ydExtendedApi均有效，区别在于ydApi里面只有静态数据，而ydExtendedApi除了同样拥有静态数据之外，还与柜台端一样管理了资金、持仓、报单和成交等动态数据，因此，在ydExtendedApi上查询资金、持仓、报单和成交等也是在本地执行的。

静态数据是指在一个交易日内不会发生改变的数据，如合约、日初保证金率、日初手续费率、账户设置等，每一种日初数据都会有其对应的结构体存放，例如合约数据对应YDInstrument、账户数据对应YDAccount。由于某些结构体同时存在静态数据和动态数据，例如行情结构体YDMarketData中的昨结算价PreSettlementPrice是静态数据，但最新价LastPrice是动态数据，而在收取静态数据阶段，易达柜台只会下发静态数据而不保证结构体中动态数据部分的正确性，因此，在收取静态数据阶段务必只使用各结构体中的静态数据而不要使用动态数据，否则可能会发生未知的错误。各个数据结构体中的静态和动态部分会在下文分别指出。如未特别说明，则对应于这些静态数据的整个结构体内的所有字段都是静态的。

> 随着日初数据量大幅增加，其中大商所传统组合持仓定义增加尤其快速，导致接收静态数据的耗时大大增加。为了加快传输速度，在1.386版本中，当1.386版本的API接入到1.386版本的柜台后，柜台会发送经过压缩的日初数据，从而大大减少传输时间。因此，若希望加快静态数据的接收速度，请将API升级到1.386版本。

登录成功以后，柜台就开始推送静态数据，API接收完成后即建立行情连接和XTCP连接（如果启用的话），无论是否连接成功，都会回调notifyFinishInit以通知策略程序所有静态数据已加载完成，这给予投资者一个时间点来为后续接收报单、成交回报做好准备，包括获取日初资金、持仓、保证金率、费率等信息。

```
virtual void notifyFinishInit(void)
```

除了通过回调函数知道静态数据推送完成外，易达API还提供了同步查询函数来查询静态数据是否推送完成，如果不希望在静态函数接收阶段收到回调消息，那么可以使用本函数判断当前初始化数据完成与否。

```
virtual bool hasFinishedInit(void)
```

下将逐一介绍易达的静态数据，以下内容中涉及到的函数在notifyFinishedInit回调中以及之后就可以正常使用。

3.2.3.1 交易所

易达目前支持中金所、上期所、能源所、大商所、广期所、郑商所、上交所（期权）、深交所（期权）共8个交易所，也支持在一套易达柜台上同时配置所有交易所连接，常见的是上期所和能源所配置在一套易达柜台上，因此，易达提供了遍历和查找交易所的两组方法，如下所示：

```
virtual int getExchangeCount(void)
virtual const YDExchange *getExchange(int pos)
virtual const YDExchange *getExchangeByID(const char *exchangeID)
```

前两个方法可以遍历所有交易所，具体方法如下面代码所示：

```
for(int i = 0; i < pApi->getExchangeCount(); i++) {
  YDExchange *exchange = pApi->getExchange(i);
}
```

最后一个方法可以指定查找一个交易所，查找各个交易所的调用如下所示：

```
pApi->getExchangeByID("CFFEX")  //中金所
pApi->getExchangeByID("SHFE")       //上期所
pApi->getExchangeByID("INE")        //能源所
pApi->getExchangeByID("DCE")        //大商所
pApi->getExchangeByID("GFEX")       //广期所
pApi->getExchangeByID("CZCE")       //郑商所
pApi->getExchangeByID("SSE")        //上交所
pApi->getExchangeByID("SZSE")       //深交所
```

YDExchange结构体中所有字段均为静态数据，大部分用于表示是否支持某功能，因此，如需详细了解具体某一字段的含义和使用方法，请在本文档中搜索该字段，以获取与该功能有关的完整介绍。

3.2.3.2 产品

易达提供了遍历和查找产品的两组方法。

```
virtual int getProductCount(void)
virtual const YDProduct *getProduct(int pos)
virtual const YDProduct *getProductByID(const char *productID)
```

前两个方法可以遍历所有产品，若此时易达柜台上配置了多个交易所，则会遍历所有交易所的所有产品，具体方法如下面代码所示：

```
for(int i = 0; i < pApi->getProductCount(); i++) {
  YDProduct *product = pApi->getProduct(i);
}
```

最后一个方法可以指定查找一个产品，查找产品的调用示例如下所示，如需知道易达所有产品的表达式，则请使用上述方法遍历后逐个查看YDProduct.ProductID：

```
pApi->getProductByID("cu")
pApi->getProductByID("cu_o")
pApi->getProductByID("IC")
```

YDProduct结构体中所有字段均为静态数据，大部分字段与YDInstrument是重复的，其中也包含了原本应该属于合约的属性，例如Multiple、Tick、UnderlyingMultiply、MaxMarketOrderVolume、MinMarketOrderVolume、MaxLimitOrderVolume、MinLimitOrderVolume，其原意是当合约上相应字段没有赋值时可以作为其默认值使用，但是目前所有合约都有妥善设置其属性值，并不会出现空值的情况，因此YDProduct上的合约属性就没有太大意义。如需了解字段含义请参考合约章节相关内容。

m_pMarginProduct用于计算大边保证金和跨产品大边保证金，详情请参见保证金优惠。

3.2.3.3 合约

易达提供了遍历和查找合约的两组方法。

```
virtual int getInstrumentCount(void)
virtual const YDInstrument *getInstrument(int pos)
virtual const YDInstrument *getInstrumentByID(const char *instrumentID)
```

前两个方法可以遍历所有合约，若此时易达柜台上配置了多个交易所，则会遍历所有交易所的所有合约，具体方法如下面代码所示：

```
for(int i = 0; i < pApi->getInstrumentCount(); i++) {
  YDInstrument *instrument = pApi->getInstrument(i);
}
```

最后一个方法可以指定查找一个合约，查找合约的调用示例如下所示，如需知道易达所有合约的表达式，则请使用上述方法遍历后逐个查看YDInstrument.InstrumentID：

```
pApi->getInstrumentByID("cu2208")
```

YDInstrument是系统的核心数据结构，除了AutoSubscribed和UserSubscribed外其他都是静态数据，下面将介绍其关键部分字段含义：

字段	说明
InstrumentID	合约代码，如cu2208，SP c2211&c2301
InstrumentHint	合约提示信息，适用于上交所/深交所ETF期权，示例：510050C2009M02350
ProductClass	合约所属产品类型 YD_PC_Futures=1：期货 YD_PC_Options=2：期权 YD_PC_Combination=3：组合，如SP c2211&c2301，目前支持大商所和郑商所的套利合约 YD_PC_Index=4：指数，如沪深300指数 YD_PC_Cash=5：现货，如沪深300ETF
DeliveryYear	交割年份，例如2022的整数
DeliveryMonth	交割月份，例如1代表1月份，12代表12月份
ExpireDate	合约到期日，例如20220808的整数
ExpireTradingDayCount	当前交易日距离合约到期日的交易日天数（不含周末和假期），由于该剩余天数的计算依赖于交易日历，因此对于到期日在明年的合约，只有当年底交易所正式发布了明年的交易日历以后才能计算正确，否则计算所需的交易日历是易达自行推算的，可能与根据正式交易日历计算出来的剩余天数有差异 如果是最后交易日，这个值是0

字段	说明
Multiple	合约乘数，相对于底层资产的倍率
Tick	最小变动价位
MaxMarketOrderVolume	市价报单的最大报单量
MinMarketOrderVolume	市价报单的最小报单量。报单量必须是其倍数
MaxLimitOrderVolume	限价报单的最大报单量
MinLimitOrderVolume	限价报单的最小报单量。报单量必须是其倍数
OptionsType	期权类型 YD_OT_NotOption=0：非期权 YD_OT_CallOption=1：看涨期权 YD_OT_PutOption=2：看跌期权
StrikePrice	行权价格，只有当合约是期权时有效
m_pUnderlyingInstrument	期权合约的基础合约的指针，可以以此获取基础合约的相关信息，只有当合约是期权时有效
UnderlyingMultiply	基础乘数，期权相对于其基础合约的倍率，只有当合约是期权时有效 对于其他非期权合约始终为1
m_pMarketData	指向行情结构体的指针，可以以此获取行情，这个行情结构体会随着行情持续刷新
m_pLegInstrument[2]	只适用于组合合约。m_pLegInstrument[0]指向组合合约中左腿合约，m_pLegInstrument[1]指向组合合约中右腿合约
LegDirction[2]	只适用于组合合约。LegDirction[0]表示组合合约中左腿合约的买卖方向，LegDirction[1]表示组合合约中右腿合约的买卖方向
m_pCombPositionDef[2][YD_MaxHedgeFlag]	只适用于组合合约。m_pCombPositionDef[0]指向买组合合约时各投保标志的组合定义数组，m_pCombPositionDef[1]指向卖组合合约时各投保标志的组合定义数组。投标标志需要减1转化为数组下标

假设某期货的合约乘数Multiply为5，表示每手期货合约对应5单位的底层资产。若每手该期货的期权对应于期货的两手，则该期权的UnderlyingMultiply为2，那么该期权的合约乘数Multiply为$5 \times 2 = 10$，表示每手期权对应10单位的底层资产。

3.2.3.4 传统组合持仓定义

在传统保证金模型中，大商所、广期所、郑商所、上交所、深交所的传统组合持仓定义可以通过以下两种API查询。

第一种方法为遍历所有的传统组合持仓定义，通过getCombPositionDefCount获取所有传统组合持仓定义的个数n，然后使用getCombPositionDef从0到n-1逐一遍历所有的传统组合持仓定义。

```
// 通过序号遍历所有传统组合持仓定义
virtual int getCombPositionDefCount(void)
virtual const YDCombPositionDef *getCombPositionDef(int pos)
```

第二种方法为通过传统组合持仓定义名称和组合投保标志查询某个传统组合持仓定义。传统组合持仓定义名称combPositionID与交易所的定义并不完全一致，请参考下文对传统组合持仓定义名称和组合投保标志的定义。

```
// 通过传统组合持仓定义名称和投保标志查询某个传统组合持仓定义
virtual const YDCombPositionDef *getCombPositionDefByID(const char *combPositionID,int combHedgeFlag)
```

通过上述API获取的YDCombPositionDef的信息如下所示：

声明	说明
YDInstrumentID CombPositionID	传统组合持仓定义名称，举例如下： pg2302,-eg2303 c2207-C-2240,-c2207-C-2240 20008571,-20008572
int CombPositionRef	传统组合持仓定义内部序号
int ExchangeRef	交易所内部序号
int Priority	传统组合持仓定义优先级，数字越小优先级越高。 可能是-1，表示该交易所的传统组合持仓不使用优先级管理，例如郑商所
short CombHedgeFlag	传统组合持仓定义投保标志，可以是以下类型： YD_CHF_SpecSpec：投机-投机 YD_CHF_SpecHedge：投机-套保 YD_CHF_HedgeHedge：套保-套保 YD_CHF_HedgeSpec：套保-投机
short CombPositionType	传统组合持仓定义类型。各个交易所的传统组合持仓定义类型如下表所示。
double Parameter	参数，不同交易所的含义不同。 目前仅用于大商所和广期所的期权对锁、买入期权垂直套利、买入期权期货组合，表示组合保证金折扣，0.2表示组合后的保证金是原两腿保证金总和的20%
const YDExchange *m_pExchange	所属YDExchange的指针
const YDInstrument *m_pInstrument[2]	传统组合持仓定义两腿合约。 两条腿的次序不一定与CombPositionID的次序相同，易达会按照业务处理的必要自行排序
int PositionDirection[2]	对应于m_pInstrument中的两腿的持仓方向
int HedgeFlag[2]	对应于m_pInstrument中的两腿的投保标志
int PositionDate[2]	对应于m_pInstrument中的两腿的持仓日期，目前全部为历史仓

各个交易所的传统组合持仓类型如下表所示。

交易所	传统组合持仓类型	说明
大商所	YD_CPT_DCE_FuturesOffset=0	期货对锁
大商所	YD_CPT_DCE_OptionsOffset=1	期权对锁
大商所	YD_CPT_DCE_FuturesCalendarSpread=2	期货跨期
大商所	YD_CPT_DCE_FuturesProductSpread=3	期货跨品种
大商所	YD_CPT_DCE_BuyOptionsVerticalSpread=4	买入期权垂直套利
大商所	YD_CPT_DCE_SellOptionsVerticalSpread=5	卖出期权垂直套利
大商所	YD_CPT_DCE_OptionsStraddle=7	卖出期权跨式
大商所	YD_CPT_DCE_OptionsStrangle=8	卖出期权宽跨式

交易所	传统组合持仓类型	说明
大商所	YD_CPT_DCE_BuyOptionsCovered=9	买入期权期货组合
大商所	YD_CPT_DCE_SellOptionsCovered=10	卖出期权期货组合
广期所	YD_CPT_GFEX_FuturesOffset=0	期货对锁
广期所	YD_CPT_GFEX_OptionsOffset=1	期权对锁
广期所	YD_CPT_GFEX_FuturesCalendarSpread=2	期货跨期
广期所	YD_CPT_GFEX_FuturesProductSpread=3	期货跨品种
广期所	YD_CPT_GFEX_BuyOptionsVerticalSpread=4	买入期权垂直套利
广期所	YD_CPT_GFEX_SellOptionsVerticalSpread=5	卖出期权垂直套利
广期所	YD_CPT_GFEX_OptionsStraddle=7	卖出期权跨式
广期所	YD_CPT_GFEX_OptionsStrangle=8	卖出期权宽跨式
广期所	YD_CPT_GFEX_BuyOptionsCovered=9	买入期权期货组合
广期所	YD_CPT_GFEX_SellOptionsCovered=10	卖出期权期货组合
中金所	YD_CPT_CFFEX_OptionCalendar=30	日历价差
中金所	YD_CPT_CFFEX_BearCall=31	熊市看涨
中金所	YD_CPT_CFFEX_BullPut=32	牛市看跌
中金所	YD_CPT_CFFEX_BullCall=33	牛市看涨
中金所	YD_CPT_CFFEX_BearPut=34	熊市看跌
郑商所	YD_CPT_CZCE_Spread=50	套利
郑商所	YD_CPT_CZCE_StraddleStrangle=51	卖出跨式或宽跨式
郑商所	YD_CPT_CZCE_SellOptionConvered=52	卖出期权期货组合
上交所、深交所	YD_CPT_StockOption_CNSJC=100	认购牛市价差
上交所、深交所	YD_CPT_StockOption_CXSJC=101	认购熊市价差
上交所、深交所	YD_CPT_StockOption_PNSJC=102	认沽牛市价差
上交所、深交所	YD_CPT_StockOption_PXSJC=103	认沽熊市价差
上交所、深交所	YD_CPT_StockOption_KS=104	跨式空头
上交所、深交所	YD_CPT_StockOption_KKS=105	宽跨式空头

3.2.3.5 日初行情

要获取日初行情，需要通过合约的m_pMarketData指针访问行情结构体YDMarketData。

YDMarketData中大部分都是动态数据，只有与上一交易日有关的字段是静态数据，静态数据部分如下表所示：

字段	说明
TradingDay	当前交易日

字段	说明
PreSettlementPrice	昨结算价
PreClosePrice	昨收盘价
PreOpenInterest	昨持仓量
UpperLimitPrice	涨停板价
LowerLimitPrice	跌停板价

3.2.3.6 账户

对于普通投资者，易达系统只提供了一种获取账户的方式，该方法只能由投资者使用，管理员用户不可以使用该方法。

```
// ydApi和ydExtendedApi均可调用
virtual const YDAccount *getMyAccount(void)

// 只有ydExtendedApi可调用
virtual const YDExtendedAccount *getExtendedAccount(const YDAccount *pAccount=NULL)
```

YDAccount和YDExtendedAccount中的静态数据如下表所示：

字段	说明
AccountRef	资金账户序号
AccountID	资金账户
PreBalance	昨余额
WarningLevel1	风险度报警级别1。已无实际作用，请忽略
WarningLevel2	风险度报警级别2。已无实际作用，请忽略
AccountFlag	账户功能开关

AccountFlag是一个位图，每一个二进制位置1表示启用该功能，置0表示关闭该功能。假设投资者开通了：柜台回报和检查报单号三个功能，那么他的AccountFlag的十进制数为80，对应的二进制数为0b1010000。

可以通过以下代码检查是否打开YD_AF_NotifyOrderAccept功能：

```
if (myAccount->AccountFlag & YD_AF_NotifyOrderAccept) {
  // server notification enabled
}
```

账户层面可以控制的功能清单如下：

开关值	说明
YD_AF_SelectConnection	是否可以把席位优选结果上传到柜台供所有人使用，详情参见席位优选结果上报。
YD_AF_AutoMakeCombPosition	**不建议使用。**是否由经纪商的ydAutoCP帮助组合，详情参见自动工具。
YD_AF_BareProtocol	是否允许使用裸协议报单，详情参见裸协议。
YD_AF_DisableSelfTradeCheck	是否关闭自成交检查，详情参见自成交检查。

开关值	说明
YD_AF_NotifyOrderAccept	是否需要柜台发送柜台回报，详情参见报单回报。
YD_AF_OrderRefCheck	是否检查OrderRef的单调递增性，详情参见报单号单调递增检查。
YD_AF_UsePositionProfit	是否将持仓盈利计入可用资金。无论是否设置，持仓浮亏始终计入可用。
YD_AF_RelaxSelfTradeCheckForQuote	放松中金所顶最后一单报价单的自成交检查逻辑，详情参见报价顶单。

3.2.3.7 日初衍生品持仓

衍生品持仓即为期货、期权的持仓，由于日初持仓的基本用法就是遍历，以便基于日初持仓计算实时持仓，因此易达只提供了遍历所有日初持仓的方法。

```
virtual int getPrePositionCount(void)
virtual const YDPrePosition *getPrePosition(int pos)
```

遍历所有持仓的方法如下所示：

```
for(int i = 0; i < pApi->getPrePositionCount(); i++) {
  YDPrePosition *prePosition = pApi->getPrePosition(i);
}
```

衍生品日初持仓的结构如下所示：

字段	说明
m_pAccount	持仓所属账户
m_pInstrument	持仓合约
PositionDirection	持仓方向
HedgeFlag	投保标志
PrePosition	日初持仓量
PreSettlementPrice	昨结算价
AverageOpenPrice	日初开仓均价

3.2.3.8 日初现货持仓

现货日初持仓包括沪深交易所的股票、基金、债券等持仓，由于日初持仓的基本用法就是遍历，以便基于日初持仓计算实时持仓，因此易达只提供了遍历所有日初持仓的方法。

```
virtual int getPreHoldingCount(void)
virtual const YDPreHolding *getPreHolding(int pos)
```

遍历所有持仓的方法如下所示：

```
for(int i = 0; i < pApi->getPreHoldingCount(); i++) {
  YDPreHolding *prePosition = pApi->getPreHolding(i);
}
```

现货日初持仓的结构如下所示：

字段	说明
m_pAccount	持仓所属账户
m_pInstrument	持仓合约
PreHolding	日初持仓量
AverageOpenPrice	日初开仓均价

3.2.3.9 日初股票期权现货持仓

股票期权现货日初持仓为作为股票期权标的物的现货持仓，由于日初持仓的基本用法就是遍历，以便基于日初持仓计算实时持仓，因此易达只提供了遍历所有日初持仓的方法。

```
virtual int getSpotPrePositionCount(void)
virtual const YDSpotPrePosition *getSpotPrePosition(int pos)
```

遍历所有持仓的方法如下所示：

```
for(int i = 0; i < pApi->getSpotPrePositionCount(); i++) {
  YDSpotPrePosition *prePosition = pApi->getSpotPrePosition(i);
}
```

股票期权现货日初持仓的结构如下所示，各个字段的计算方式参见股票期权现货持仓模型：

字段	说明
m_pAccount	持仓所属账户
m_pInstrument	持仓合约
Position	日初持仓量
ExchangeFrozenVolume	交易所冻结数量
ExecAllocatedVolume	净行权分配量，正值代表收券，负值代表交券，仅E+1日有意义
ExecAllocatedAmount	净行权分配金额，正值代表收钱，负值代表交钱，仅E+1日有意义
ExecAllocatedFrozenVolume	行权分配冻结数量，仅E+1日有意义
ExecAllocatedFrozenAmount	行权分配冻结金额，仅E+1日有意义

3.2.3.10 日初传统组合持仓

在传统保证金模型中，日初传统组合持仓是静态数据，理论上它也应该在静态数据传输阶段，也就是notifyFinishInit之前传输，但是为了兼容老版本API，实际上它是紧跟在notifyFinishInit之后、notifyCaughtUp之前传输到客户端的，虽然传输时间点与其他静态数据不同，但是与其他静态数据一样，日初传统组合持仓只会在首次登录时推送一次，也就是有notifyFinishInit回调时会被推送到客户端，不会因为断线等原因导致重复推送。为了概念的正确性和便于理解，日初传统组合持仓的介绍与其他静态数据放在一起。

API没有提供查询接口，因此只能从notifyCombPosition回调函数中收集所有日初传统组合持仓。

```
virtual void notifyCombPosition(const YDCombPosition *pCombPosition,const YDCombPositionDef *pCombPositionDef,const YDAccount *pAccount)
```

YDCombPosition结构体里面的字段都是静态数据，使用时需要结合该回调中其他参数一起使用：

参数	字段	说明
pAccount		持仓所属账户
pCombPositionDef		传统组合持仓定义
pCombPosition	Position	持仓量
	CombPositionDetailID	传统组合持仓明细ID，只对上交所、深交所有意义

3.2.3.11 系统参数

系统参数是在计算业务过程中会使用到的参数，目前只用于保证金计算。

易达提供了遍历和查找的两组方法。

```
virtual int getSystemParamCount(void)
virtual const YDSystemParam *getSystemParam(int pos)
virtual const YDSystemParam *getSystemParamByName(const char *name,const char *target)
```

前两个方法可以遍历所有系统参数，具体方法如下面代码所示：

```
for(int i = 0; i < pApi->getSystemParamCount(); i++) {
  YDSystemParam *param = pApi->getSystemParam(i);
}
```

第三个方法可以指定查找系统参数，查找调用示例如下所示：

```
getSystemParamByName("MarginBasePrice", "Futures")
```

YDSystemParam都是静态数据，其结构如下：

字段	说明
Name	参数名称
Target	参数的适用对象
Value	参数值

上述Name和Target都是字符串，可能的搭配组合如下表所示：

Name	Target	Value
MarginLowerBoundaryCoef	MarginCalcMethod1	股指期权保证金算法中的保障系数，浮点数，缺省值0.5
MarginBasePrice	Futures或者Options	计算期货或者期权空头的保证金时使用的基础价格，可以选择的值如下： 0：昨结算价 1：开仓价 2：最新价 3：市场平均成交价 4：最新价和昨结算价之间的较大值 目前的常用规则是期货为1，期权为4

Name	Target	Value
OrderMarginBasePrice	Futures或者Options	计算期货或者卖出期权的开仓报单的冻结保证金时使用的基础价格，可以选择的值如下： 0：昨结算价 5：报单价（对于市价单，使用涨停板） 7: 使用MarginBasePrice的价格 目前的常用规则是期货为5，期权为0 注意，期权执行申请报单中冻结的保证金永远是基于昨结算价的
MarginBasePriceAsUnderlying	Options	计算卖出期权保证金时使用的基础产品的价格，可以选择的值如下： 0：昨结算价 2：最新价 4：最新价和昨结算价之间的较大值 目前的常用规则是0
SellOrderPremiumBasePrice	Options	计算卖出开仓期权时反向冻结权利金使用的基础价格，可以选择的值如下： 0：昨结算价 5：报单价（对于市价单，使用跌停板） 6：不反向冻结 目前的常用规则是6 注意，买入开仓期权时冻结的权利金永远是报单价（对于市价单，使用涨停板）
PortfolioMarginConcession	DCELongOptionPortfolio	对于大商所和广期所包含期权多仓的传统组合持仓，是否优惠计算保证金。可以选择的值如下： 0：不优惠 1：优惠。由于易达平仓时不会检查资金，因此极端情况下可能会导致买期权组合平仓时多收保证金而造成资金透支。
UseCollateral	Account	将质押品获得的质押资金计入昨权益供开仓使用。可以选择的值如下： 0：关闭 1：启用
MarketMaker	System	柜台是否支持做市商报价业务 yes：支持 no：不支持
Test	System	柜台是否是测试柜台，测试柜台不包含性能优化，不可以用于生产
ServerVersion	System	柜台版本号，版本号规范详见版本规则
ConnectionMode	System	柜台的席位模式： 0：全管理席位 1：首席位管理其他非管理 2：全非管理席位 3：全管理作为非管理席位使用，只能用于上期所和中金所
MinApiVersion	System	柜台支持的最低API版本号，版本号规范详见版本规则

Name	Target	Value
MaxApiVersion	System	柜台支持的最高API版本号，版本号规范详见版本规则
MinSelectConnectionGap	System	上报席位优选结果的最小时间间隔，单位为毫秒
MaxSelectConnectionGap	System	上报席位优选结果的最大有效时间，单位为毫秒
MissingOrderGap	System	被系统判定为超时未知单的超时时间，单位为毫秒。
UDPTradingPort	System	UDP报单端口。若柜台没有开启UDP服务，则为0。
XTCPTradingPort	System	XTCP报单端口。若柜台没有开启XTCP服务，则为0。
TradingSegmentDetail	System	是否开启了详细的交易阶段通告 yes：开启 no：关闭
ClientMinPasswordLen	System	投资者最小口令长度 0表示没有最小长度限制
ClientMinPasswordCharSet	System	投资者口令字符集数量 0表示没有字符集数量限制
AdminMinPasswordLen	System	管理员最小口令长度 0表示没有最小长度限制
AdminMinPasswordCharSet	System	管理员口令字符集数量 0表示没有字符集数量限制

3.2.3.12 现货交易规费率

> 现货交易规费率的设置值是稀疏的，例如要设置全体投资者的产品层面的现货交易规费率，只需要在产品层面设置一条记录即可，易达系统会将这条设置值应用到所有投资者的所有属于该产品的合约上。这些设置值主要供易达界面显示所用，对普通投资者没有实际的意义。
> 下面是获取费率设置值的方法，仅供参考，不做详细解释。

```
// 手续费率设置值
virtual int getCashCommissionRateCount(void)
virtual const YDCashCommissionRate *getCashCommissionRate(int pos)
```

投资者可以使用getInstrumentCashCommissionRate获取合约在对应业务和买卖方向上的现货交易规费率，不建议直接从账户层级信息中的YDAccountInstrumentInfo读取。

```
virtual const YDCashCommissionRate *getInstrumentCashCommissionRate(const YDInstrument
*pInstrument,int ydOrderFlag,int direction,const YDAccount *pAccount=NULL)
```

返回的现货交易规费率YDCashCommissionRate的字段说明如下所示：

字段	说明
SubProductClass	YD_SPC_Other=0：其他 YD_SPC_Stock=1：股票 YD_SPC_Bond=2：基金 YD_SPC_Fund=3：债券

字段	说明
YDOrderFlag	YD_YOF_Normal=0：普通交易 YD_YOF_Designation=11：深交所转托管
Direction	YD_D_Buy=0：买入 YD_D_Sell=1：卖出
RatePiece[YD_CCT_Count]	交易规费细项数组，每个数组元素即为一种费用类型YDCashCommissionRatePiece。 YD_CCT_StampDuty：印花税率 YD_CCT_SecuritiesManagementFee：证管费率 YD_CCT_HandlingFee：经手费率 YD_CCT_TransferFee：过户费率
m_pInstrument	设置值指向的合约
m_pProduct	设置值指向的产品
m_pExchange	设置值指向的交易所
m_pAccount	设置值指向的资金账户

> **Note**
>
> m_pInstrument、m_pProduct、m_pExchange、m_pAccount指向的是当前费率结构体设置的层级，具体合约的费率结构体中的这些字段通常与该合约无关，因此投资者通常无需关注这些字段的值。例如，某项设置在上交所（m_pExchange指向上交所）的费率会应用到上交所所有证券上，所有上交所证券均指向相同的结构体，而这些结构体的m_pExchange均为上交所指针，与这些合约没有关系。

YDCashCommissionRatePiece的结构说明如下所示：

字段	说明
RateByAmount	以金额计算的费率
RateByVolume	以数量计算的费率
MaxValue	最大费用金额
MinValue	最小费用金额

3.2.3.13 现货佣金费率

> 现货佣金费率的设置值是稀疏的，例如要设置全体投资者的产品层面的现货佣金费率，只需要在产品层面设置一条记录即可，易达系统会将这条设置值应用到所有投资者的所有属于该产品的合约上。这些设置值主要供易达界面显示所用，对普通投资者没有实际的意义。
> 下面是获取费率设置值的方法，仅供参考，不做详细解释。
>
> ```
> // 手续费率设置值
> virtual int getBrokerageFeeRateCount(void)
> virtual const YDBrokerageFeeRate *getBrokerageFeeRate(int pos)
> ```

投资者可以使用getInstrumentBrokerageFeeRate获取合约在对应业务和买卖方向上的现货佣金费率，不建议直接从账户层级信息中的YDAccountInstrumentInfo读取。

```
virtual const YDBrokerageFeeRate *getInstrumentBrokerageFeeRate(const YDInstrument
*pInstrument,int ydOrderFlag,int direction,const YDAccount *pAccount=NULL)
```

返回的现货佣金费率YDBrokerageFeeRate的字段说明如下所示：

字段	说明
SubProductClass	YD_SPC_Other=0：其他 YD_SPC_Stock=1：股票 YD_SPC_Bond=2：基金 YD_SPC_Fund=3：债券
YDOrderFlag	YD_YOF_Normal=0：普通交易 YD_YOF_Designation=11：深交所转托管
Direction	YD_D_Buy=0：买入 YD_D_Sell=1：卖出
RatePiece	指向现货佣金费率的明细参数，YDCashCommissionRatePiece请参考现货交易规费率
m_pInstrument	设置值指向的合约
m_pProduct	设置值指向的产品
m_pExchange	设置值指向的交易所
m_pAccount	设置值指向的资金账户

> **Note**
>
> m_pInstrument、m_pProduct、m_pExchange、m_pAccount指向的是当前费率结构体设置的层级，具体合约的费率结构体中的这些字段通常与该合约无关，因此投资者通常无需关注这些字段的值。例如，某项设置在上交所（m_pExchange指向上交所）的费率会应用到上交所所有证券上，所有上交所证券均指向相同的结构体，而这些结构体的m_pExchange均为上交所指针，与这些合约没有关系。

3.2.3.14 手续费率

> 费率的设置值是稀疏的，例如要设置全体投资者的产品层面的费率，只需要在产品层面设置一条记录即可，易达系统会将这条设置值应用到所有投资者的所有属于该产品的合约上。这些设置值主要供易达界面显示所用，对普通投资者没有实际的意义。
> 下面是获取费率设置值的方法，仅供参考，不做详细解释。
>
> ```
> // 手续费率设置值
> virtual int getCommissionRateCount(void)
> virtual const YDCommissionRate *getCommissionRate(int pos)
> ```

投资者可以使用getInstrumentCommissionRate获取合约在对应的投保标志上的手续费率，不建议直接从账户层级信息中的YDAccountInstrumentInfo读取。

```
virtual const YDCommissionRate *getInstrumentCommissionRate(const YDInstrument
*pInstrument,int hedgeFlag,const YDAccount *pAccount=NULL)
```

返回的手续费率的对应关系如下表所示：

参数	字段	说明
YDCommissionRate	OpenRatioByMoney	开仓成交金额手续费率
	OpenRatioByVolume	开仓成交每手手续费
	CloseRatioByMoney	平昨仓成交金额手续费率

参数	字段	说明
	CloseRatioByVolume	平昨仓成交每手手续费
	CloseTodayRatioByMoney	平今仓成交金额手续费率
	CloseTodayRatioByVolume	平今仓成交每手手续费
	OrderCommByVolume	报单每笔手续费
	OrderActionCommByVolume	撤单每笔手续费
	ExecRatioByMoney	行权金额手续费率
	ExecRatioByVolume	行权每手手续费

3.2.3.15 信息量申报费率

信息量申报费率包含了各交易所申报费收取标准，由于部分交易所的部分产品并没有开始征收信息量申报费，因此，以下方法获取的仅包含收取了信息量产品的参数信息：

```
// 信息量申报费率设置值
virtual int getMessageCommissionRateCount(void)
virtual const YDMessageCommissionRate *getMessageCommissionRate(int pos)
```

其中，YDMessageCommissionRate的字段信息如下：

字段	说明
ProductRef	产品Ref。可以参考YDProduct类获取。
MessageCount	信息量档位
OTR	OTR档位
CommissionRate	信息量申报费率

交易所公布的OTR和信息量是区间值，而易达中提供的是档位参数，两者的对应关系如以下两张表格所示。

以2022年上期所公布的铜期货cu的信息量申报费率为例：

	$OTR \leq 2$	$OTR > 2$
$1笔 \leq 信息量 \leq 4000笔$	0元/笔	0元/笔
$4001笔 \leq 信息量 \leq 8000笔$	0.25元/笔	0.5元/笔
$4001笔 \leq 信息量 \leq 8000笔$	1.25元/笔	2.5元/笔
$40001笔 \leq 信息量$	25元/笔	50元/笔

以下为易达对应的设置，假设铜期货cu对应的ProductRef为8：

ProductRef	OTR	MessageCount	CommissionRate
8	0	0	0
8	0	4000	0.25
8	0	8000	1.25

ProductRef	OTR	MessageCount	CommissionRate
8	0	40000	25
8	2	0	0
8	2	4000	0.5
8	2	8000	2.5
8	2	40000	50

假设某投资者在cu某合约上的信息量为9000，对应的OTR为3，那么按照分段累计方式计算的信息量申报费为：

- 第一段：共4000张，申报费为0元
- 第二段：共4000张，申报费为4000*0.5=2000元
- 第三段：共4000张，申报费为1000*2.5=2500元
- 信息量申报费共计4500元

3.2.3.16 传统保证金参数

> 传统保证金率的设置值是稀疏的，例如要设置全体投资者的产品层面的传统保证金率，只需要在产品层面设置一条记录即可，易达系统会将这条设置值应用到所有投资者的所有属于该产品的合约上。这些设置值主要供易达界面显示所用，对普通投资者没有实际的意义。
> 下面是获取传统保证金率设置值的方法，仅供参考，不做详细解释。

```
// 保证金率设置值
virtual int getMarginRateCount(void)
virtual const YDMarginRate *getMarginRate(int pos)
```

投资者可以使用getInstrumentMarginRate获取合约在对应的投保标志上的保证金率，不建议直接从账户层级信息中的YDAccountInstrumentInfo读取。

```
virtual const YDMarginRate *getInstrumentMarginRate(const YDInstrument *pInstrument,int hedgeFlag,const YDAccount *pAccount=NULL)
```

返回的YDMarginRate结构体使用了大量的union字段，其目的是为了适应不同的保证金模型，为方便理解，下面将分别列出三种不同保证金模型下的保证金率参数。

以下字段适用于期货交易所的期货保证金。

字段	说明
m_pAccount	保证金率所属投资者的指针
m_pProduct	保证金率所属产品的指针
m_pInstrument	保证金率对应的合约的指针
HedgeFlag	YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。为便于编写程序，易达为是否支持套利交易和持仓提供了参数化表示，YDExchange.UseArbitragePosition为true时表示该交易所支持套利交易和持仓，否则为不支持 YD_HF_Hedge=3：套保
LongMarginRatioByMoney	基于金额的多头保证金率
LongMarginRatioByVolume	基于手数的多头保证金率

字段	说明
ShortMarginRatioByMoney	基于金额的空头保证金率
ShortMarginRatioByVolume	基于手数的空头保证金率

以下字段适用于期货交易所的商品期权保证金和中金所的股指期权保证金。

字段	说明
m_pAccount	保证金率所属投资者的指针
m_pProduct	保证金率所属产品的指针
m_pInstrument	保证金率对应的合约的指针
HedgeFlag	YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。为便于编写程序，易达为是否支持套利交易和持仓提供了参数化表示，YDExchange.UseArbitragePosition为true时表示该交易所支持套利交易和持仓，否则为不支持 YD_HF_Hedge=3：套保
CallMarginRatioByMoney	基于金额的看涨期权空头保证金率
CallMarginRatioByVolume	基于手数的看涨期权空头保证金率
PutMarginRatioByMoney	基于金额的看跌期权空头保证金率
PutMarginRatioByVolume	基于手数的看跌期权空头保证金率

以下字段适用于上交所和深交所的股票期权保证金。

字段	说明
m_pAccount	保证金率所属投资者的指针
m_pProduct	保证金率所属产品的指针
m_pInstrument	保证金率对应的合约的指针
HedgeFlag	YD_HF_Normal=1：普通 YD_HF_Covered=3：备兑
BaseMarginRate	基础合约保证金率
LinearFactor	线性系数
LowerBoundaryCoef	最低保障系数

3.2.3.16.1 盘中调保

通常情况下，保证金率在盘中交易时是固定不变了，但在某些情况下经纪商可能会在盘中对某些客户调整保证金。当经纪商更新盘中保证金率时，投资者会收到以下回调：

```
virtual void notifyUpdateMarginRate(const YDUpdateMarginRate *pUpdateMarginRate)
```

当保证金率发生改变时，使用ydExtendedApi的投资者不用关心究竟是哪些合约发生了改变以及需要更新哪些持仓的保证金，ydExtendedApi会妥善管理好保证金率变更的相关工作，但是使用ydApi的投资者就需要自主更新保证金率发生变化的合约对应的持仓保证金。下表为更新保证金率YDUpdateMarginRate中表示其影响保证金率的合约过滤字段，保证金率相关字段不予列出，需要请参考上文各个保证金模型对应的保证金率说明。

字段	说明
AccountRef	账户序号
ProductRef	产品序号
InstrumentRef	合约序号
HedgeFlagSet	投保标志集合，若某投保标志受到影响，1<<HedgeFlag对应的位会被设置
OptionTypeSet	期权类型集合，若某类型期权受到影响，1<<OptionsType对应的位会被设置
ExpireDate	合约到期日，主要用于表示期权系列
UnderlyingInstrumentRef	基础合约序号
Multiple	合约乘数，股票期权可能会受到分红影响而调整其保证金率

下面给出了当收到某次保证金率变更时，判断该变更是否会影响到某持仓（合约与投保标志）的示例代码：

```
bool applyToInstrument(const CYDUpdateMarginRate *pUpdateMarginRate,const YDInstrument *pInstrument, int hedgeFlag) const
{
    if ((pUpdateMarginRate->ProductRef>=0) && (pInstrument->ProductRef!=pUpdateMarginRate->ProductRef))
    {
        return false;
    }
    if ((pUpdateMarginRate->InstrumentRef>=0) && (pInstrument->InstrumentRef!=pUpdateMarginRate->InstrumentRef))
    {
        return false;
    }
    if ((pUpdateMarginRate->UnderlyingInstrumentRef>=0) && (pInstrument->UnderlyingInstrumentRef!=pUpdateMarginRate->UnderlyingInstrumentRef))
    {
        return false;
    }
    if ((pUpdateMarginRate->ExpireDate>0) && (pInstrument->ExpireDate!=pUpdateMarginRate->ExpireDate))
    {
        return false;
    }
    if ((pUpdateMarginRate->Multiple>0) && (pInstrument->Multiple!=pUpdateMarginRate->Multiple))
    {
        return false;
    }
    if ((pUpdateMarginRate->OptionTypeSet>0) && (((1<<pInstrument->OptionsType)&pUpdateMarginRate->OptionTypeSet)==0))
    {
        return false;
    }
  if ((pUpdateMarginRate->HedgeFlagSet>0) && (((1<<hedgeFlag)&pUpdateMarginRate->HedgeFlagSet)==0))
```

```
  {
    return false;
  }
    return true;
}
```

建议ydApi的用户遍历持仓刷新各个持仓的保证金，伪代码如下：

```
virtual void notifyUpdateMarginRate(const YDUpdateMarginRate *pUpdateMarginRate)
{
  for(auto pPosition : AllPositions)
  {
    if (applyToInstrument(pUpdateMarginRate, pPosition->pInstrument, pPosition->hedgeFlag))
    {
      // 使用投资者自己的保证金刷新方法
      userDefinedUpdateMargin(pPosition, pUpdateMarginRate);
    }
  }
}
```

3.2.3.17 账户层级信息

账户层级信息提供了交易所级、产品级、合约级的保证金率、费率、权限和传统风控参数，是专门供投资者获取这些参数的结构体，对于这几类信息的说明如下：

- 费率是静态信息，保证金是动态数据，盘中可调，因此在notifyFinishInit中获得的是日初保证金，费率和保证金率是合约层信息。为方便使用，API提供了直接获取合约层面保证金率、费率的方法，详情请参见保证金率和手续费相关内容
- 权限是动态数据，盘中可调，保证金参数在各个层面都有设置，但最终要落实到合约层面。API没直接提供查询合约层面权限的方法，可以按照交易权限中的方法自行实现
- 传统风控参数是静态参数，易达的风控参数可以设置在产品和合约两个层面，例如设置在产品层面的撤单数表示限制该产品中所有合约撤单数的总和，因此，需要分别从产品和合约层面获取风控参数。详情请参见风控参数

账户各层级信息可以直接通过下列函数获取：

```
/// when trader call following 3 functions, pAccount should be NULL
virtual const YDAccountExchangeInfo *getAccountExchangeInfo(const YDExchange *pExchange,const YDAccount *pAccount=NULL)
virtual const YDAccountProductInfo *getAccountProductInfo(const YDProduct *pProduct,const YDAccount *pAccount=NULL)
virtual const YDAccountInstrumentInfo *getAccountInstrumentInfo(const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
```

YDAccountExchangeInfo的主要字段如下所示，其主键为账户和交易所：

字段	说明
m_pAccount	账户结构体指针
m_pExchange	交易所结构体指针
TradingRight	交易权限，动态数据
IsDedicatedConnectionID	专属席位数组位图，详情参见指定席位
TradingCode[YD_MaxHedgeFlag]	交易编码数组，可以获得不同投保标志下的交易编码

YDAccountProductInfo的主要字段如下所示，其主键为账户和产品：

字段	说明
m_pAccount	账户结构体指针
m_pExchange	交易所结构体指针
TradingRight	交易权限，动态数据
TradingConstraints[YD_MaxHedgeFlag]	投机、套保、套利传统风控参数数组，表示属于该产品的所有合约的风控指标相加后予以风控，详情参见风控参数

YDAccountInstrumentInfo的主要字段如下所示，其主键为账户和合约：

字段	说明
m_pAccount	账户结构体指针
m_pExchange	交易所结构体指针
TradingRight	交易权限，动态数据
TradingConstraints[YD_MaxHedgeFlag]	投机、套保、套利传统风控参数数组，表示只限制该合约风控指标上的阈值，详情参见风控参数
m_pMarginRate[YD_MaxHedgeFlag]	保证金率数组，详情参见保证金率
m_pCommissionRate[YD_MaxHedgeFlag]	手续费率数组，详情参见手续费
m_pCashCommissionRate[2]	现货交易规费费率，m_pCashCommissionRate[0]为买现货交易规费费率，m_pCashCommissionRate[1]为卖现货交易规费费率
m_pBrokerageFeeRate[2]	现货佣金费率，m_pCashCommissionRate[0]为买现货佣金费率，m_pCashCommissionRate[1]为卖现货佣金费率
RFQCount	询价单计数，仅计算上期所和能源所的期权询价单
ExemptMessageCommissionYDOrderFlag	当报单的YDOrderFlag大于本标志时不计算信息量申报费
MaxMessage	最大信息量上限，超过上限将禁止在该合约上报单

对于使用YDExtendedApi的投资者，还可以通过以下方法获得更详细的账户信息，这些方法的入参即为上述获得的各个层级的指针。

```
1 virtual const YDExtendedAccountExchangeInfo *getExtendedAccountExchangeInfo(const
  YDAccountExchangeInfo *pAccountExchangeInfo)
2 virtual const YDExtendedAccountProductInfo *getExtendedAccountProductInfo(const
  YDAccountProductInfo *pAccountProductInfo)
3 virtual const YDExtendedAccountInstrumentInfo *getExtendedAccountInstrumentInfo(const
  YDAccountInstrumentInfo *pAccountInstrumentInfo)
```

YDExtendedAccountExchangeInfo相比YDAccountExchangeInfo提供的额外信息如下所示：

字段	说明
OptionLongPositionCost	买入额度（期权多头持仓总成本）
OptionLongPositionCostLimit	买入额度限额（期权多头持仓总成本限额）
TradeControlFlag	现货交易投资者适当性控制标志位

YDExtendedAccountProductInfo相比YDAccountProductInfo提供的额外信息如下所示：

字段	说明
MarginModelID	保证金模型ID YD_MM_SPBM=1：郑商所SPBM组合保证金模型 YD_MM_RULE=2：大商所RULE组合保证金模型 YD_MM_SPMM_SHFE=3：上期所SPMM组合保证金模型 YD_MM_SPMM_INE=4：能源所SPMM组合保证金模型 YD_MM_RCAMS=5：中金所RCAMS组合保证金模型

YDExtendedAccountInstrumentInfo相比AccountInstrumentInfo提供的额外信息如下所示：

字段	说明
MessageCommission	当前信息量手续费
CashTradingConstraint	现货交易约束位图，各位表示的含义如下： 0：禁止买入 1：禁止卖出
MaxCashBuyVolume	现货日累计买入数量限制
MaxOrderVolume	最大委托数量
HoldingLimit	现货持仓量限制

3.2.3.18 组合保证金参数

在组合保证金模型中，组合保证金参数根据交易所日初数据转换而来，盘中不可修改。为了适应所有交易所的组合保证金参数，易达采用通用表达方式，即采用键值对的表达方式，键值对的含义由各组合保证金模型自行解析。更多关于组合保证金模型的内容，请参考组合保证金模型。

易达提供了下列遍历所有保证金模型的方法：

```
virtual int getMarginModelParamCount(void)
virtual const YDMarginModelParam *getMarginModelParam(int pos)
```

其中，YDMarginModelParam的结构为：

字段	说明
MarginModelID	保证金模型ID YD_MM_SPBM=1：郑商所SPBM组合保证金模型 YD_MM_RULE=2：大商所RULE组合保证金模型 YD_MM_SPMM_SHFE=3：上期所SPMM组合保证金模型 YD_MM_SPMM_INE=4：能源所SPMM组合保证金模型 YD_MM_RCAMS=5：中金所RCAMS组合保证金模型
ParamName	参数名
ParamValue	参数值

以郑商所SPBM参数为例，返回的保证金参数如下表所示，其中郑商所SPBM的保证金模型ID是1：

MarginModelID	ParamName	ParamValue
1	MA.intraRateY	0.7
1	MA.fut.cvf	10
1	MA.fut.MA211.price	2603
1	MA.fut.MA211.marginRate	0.2
1	MA.fut.MA211.timeRange	SPOT
1	MA.fut.MA211.lockRateX	0.2
1	MA.fut.MA211.addOnRate	0

上述数据等同于以下郑商所下发原始文件中的信息：

```
<?xml version="1.0" encoding="utf-8"?>
<spbmFile>
    <version>1.00</version>
    <exchange>ZCE</exchange>
    <exchangeName>郑州商品交易所</exchangeName>
    <createdDate>20221101</createdDate>
    <createdTime>150229</createdTime>
    <productFamily>
        <productFamilyCode>MA</productFamilyCode>
        <intraRateY>70.00</intraRateY>
        <futPf>
            <pfCode>MA</pfCode>
            <pfName>甲醇期货</pfName>
            <cvf>10</cvf>
            <fut>
                <month>202211</month>
                <futCode>MA211</futCode>
                <price>2603.00</price>
                <marginRate>20.00</marginRate>
                <timeRange>SPOT</timeRange>
                <lockRateX>20.00</lockRateX>
                <addOnRate>0.00</addOnRate>
            </fut>
        </futPf>
    </productFamily>
</spbmFile>
```

3.2.3.19 账户组合保证金参数

在组合保证金模型中，可通过以下方法获得投资者在某个保证金模型上的参数。

```
virtual const YDAccountMarginModelInfo *getAccountMarginModelInfo(int marginModelID,const YDAccount *pAccount=NULL)
```

其中，YDAccountMarginModelInfo结构体的字段如下：

字段	说明
AccountRef	账户序号。可从YDAccount中获取

字段	说明
MarginModelID	保证金模型ID YD_MM_SPBM=1：郑商所SPBM组合保证金模型 YD_MM_RULE=2：大商所RULE组合保证金模型 YD_MM_SPMM_SHFE=3：上期所SPMM组合保证金模型 YD_MM_SPMM_INE=4：能源所SPMM组合保证金模型 YD_MM_RCAMS=5：中金所RCAMS组合保证金模型
CloseVerify	平仓时是否检查可用资金
MarginRatio	保证金系数
ProductRange	适用的产品范围。用空格分隔，表示适用于对应组合保证金模型的产品子集 为空表示不限制使用产品范围，与组合保证金模型一致

拥有修改保证金率权限的管理员可以在盘中修改账户组合保证金参数中的CloseVerify和MarginRatio，ProductRange不可修改。修改方法如下。

```
virtual bool adjustAccountMarginModelInfo(const YDAccountMarginModelInfo
*pAccountMarginModelInfo,int requestID=0)
```

若盘中CloseVerify或MarginRatio发生变化，会通过以下回调函数通知投资者。

```
virtual void notifyAccountMarginModelInfo(const YDAccountMarginModelInfo
*pAccountMarginModelInfo)
```

3.2.3.20 产品群组合保证金参数

从`1.486`版本开始，易达支持在产品群上设置投资者保证金系数，可通过以下方法遍历获取。

```
virtual int getAccountProductGroupMarginModelParamCount(void)
virtual const YDAccountProductGroupMarginModelParam
*getAccountProductGroupMarginModelParam(int pos)
```

遍历样例代码如下所示：

```
for(int i = 0; i < pApi->getAccountProductGroupMarginModelParamCount(); i++) {
  YDAccountProductGroupMarginModelParam *productGroupParam = pApi-
>getAccountProductGroupMarginModelParam(i);
}
```

其中，YDAccountProductGroupMarginModelParam结构体的字段如下：

字段	说明
AccountRef	账户序号。可从YDAccount中获取
MarginModelID	保证金模型ID YD_MM_SPBM=1：郑商所SPBM组合保证金模型 YD_MM_RULE=2：大商所RULE组合保证金模型 YD_MM_SPMM_SHFE=3：上期所SPMM组合保证金模型 YD_MM_SPMM_INE=4：能源所SPMM组合保证金模型 YD_MM_RCAMS=5：中金所RCAMS组合保证金模型
ProductGroupName	产品群

字段	说明
MarginRatio	产品群保证金系数

更多关于组合保证金模型的内容，请参考组合保证金模型。

3.2.3.21 风控参数

易达系统可以按照技术实现来划分为两类风控参数，分别是传统风控参数以及通用风控参数。

- 传统风控参数是易达早期支持的风控规则，主要是常见的开仓量、成交量、持仓量和撤单笔数的控制，传统风控参数不会继续增加，新的风控规则参数全部在通用风控参数中设置
- 通用风控参数是为了适应越来越灵活复杂的风控规则而引入的风控参数设置方法，其设置结构采用了通用的设置方法，要根据每个风控规则去解读其参数，与在账户层级信息中的最终值不同，通用风控参数并没有将不同维度设置的风控参数做归并，需要投资者根据每个风控规则支持的层次结构自行归并

传统风控参数结构体的字段如下表所示，与YDAccountProductInfo和YDAccountInstrumentInfo中的TradingConstraints[YD_MaxHedgeFlag]相结合，就能获得完整的不同投保标志下的风控参数，如需了解详细的风控规则计算方式请参见风控中的具体风控规则。

字段	说明
OpenLimit	开仓量限制
DirectionOpenLimit[2]	买卖开仓量限制
PositionLimit	持仓量限制
DirectionPositionLimit[2]	多空持仓量限制
TradeVolumeLimit	成交量限制
CancelLimit	撤单笔数限制

通用风控规则的参数可以通过以下遍历方法获取：

```
virtual int getGeneralRiskParamCount(void)
virtual const YDGeneralRiskParam *getGeneralRiskParam(int pos)
```

遍历全部通用风控规则参数的样例代码如下所示：

```
for(int i = 0; i < pApi->getGeneralRiskParamCount(); i++) {
  YDGeneralRiskParam *param = pApi->getGeneralRiskParam(i);
}
```

通用风控参数YDGeneralRiskParam是静态数据，其结构体如下所示，主键为(GeneralRiskParamType，AccountRef，ExtendedRef)：

字段	说明
GeneralRiskParamType	风控规则类型
AccountRef	账户序号，-1表示对所有用户生效，其他值表示对本用户生效
ExtendedRef	扩展序号，根据规则定义可以表示交易所、产品、合约的序号
IntValue1	整型参数1

字段	说明
IntValue2	整型参数2
FloatValue	浮点参数

并非所有的风控规则都使用了上面所有的IntValue1、IntValue2和FloatValue，大部分风控规则只使用了上述规则中的部分参数，请参考具体的风控规则，在风控规则中没有列出的参数值则表示没有使用。

3.2.4 收取动态数据

易达柜台在推送完成静态数据以后，柜台端已经做好了接受报单的准备，如果此时开始报单柜台就会正常推送回报等动态数据。但若此时柜台尚有客户端离线期间未发送到客户端的动态数据，那么客户此时交易的基础可能是错误的，为了通知投资者已经完成了所有历史动态数据的接收并可以开始正常交易，易达增加了这一阶段并通过回调函数notifyCaughtUp通知投资者。

```
virtual void notifyCaughtUp(void)
```

基本原理是登录的时候柜台会告诉该投资者最大的流水号，API在接收时会比较当前动态数据的流水号与登录时的最大流水号，若当前流水号超过了登录时的最大流水号，那么就说明流水的追及已经完成。该方法被回调仅说明追上了登录时的流水，实际还可能存在登录后产生的流水。如果发生了断线重连，会再次获得该消息，详情请参见交易重连。

本阶段内收取的动态数据与正常交易阶段收取的数据是一样的，因此在此就不再列举收取动态数据的回调函数，请参考对应业务中对于回调函数的说明。

3.3 断连

投资者可以使用以下方法主动断开与柜台的交易和行情TCP连接：

```
virtual void disconnect(void)
```

但是由于ydApi重连的机制，在调用上述方法成功断开连接后，ydApi会尝试重新连接柜台。因此，如果想要彻底断开与柜台的连接，请销毁ydApi。

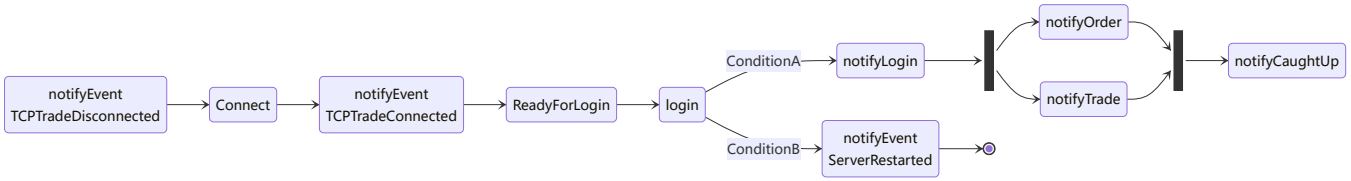
3.4 重连

3.4.1 交易重连

当API探测到与柜台的TCP交易连接中断或者超时后，API会回调notifyEvent(YD_AE_TCPTradeDisconnected)通知交易线程断开了与柜台的连接，此后，API会持续发起到柜台的连接，直到重新连接上后API会回调notifyEvent(YD_AE_TCPTradeConnected)通知，后续的过程与启动过程类似，但是有以下区别：

- 重连过程中不会重新传输静态数据，因为就不会有notifyFinishInit回调，在登录完成后就开始传输断线过程中发生的报单和成交等流水
- login时不允许换成另外的用户登录，调用login登录的过程中，API会根据诸多因素判断是否应该继续后续步骤还是终止API，具体逻辑如下：
  - 检查柜台数据指纹是否发生了变化，若发生了变化，则直接进入终止逻辑（ConditionB）。柜台数据指纹是根据日初数据和易达配置信息综合计算出来的，上传了错误的日初数据后重新上传并重启柜台、新的交易日数据被上传并重启柜台、柜台配置的交易所或者席位等技术参数发生了改变并重启柜台，都是常见的导致柜台数据指纹发生变化的原因
  - 如果柜台数据指纹没有变化，则需要继续检查柜台实例ID。每次启动的柜台实例ID都不相同，因此柜台实例ID可以用于判断柜台是否发生了重启。根据柜台是否重启以及API是否启用了高可用功能，存在以下三种可能性：
    - 若柜台没有发生重启，说明此次断线是因为网络问题导致的，系统将正常继续执行后续步骤（ConditionA）

    - 若柜台发生了重启，并且API启用了高可用功能，则会继续正常执行后续步骤（ConditionA），此时在notifyLogin之前会收到notifyEvent(YD_AE_ServerSwitch)回调，表示发生了一次主备切换
    - 若API没有启用高可用功能，则会直接进入终止逻辑（ConditionB）
  - 进入终止逻辑后，通常情况下API会调用exit()系统调用导致整个进程退出，这会连带策略程序一起退出。如果希望避免这种情况，可以在notifyEvent(YD_AE_ServerRestarted)中主动销毁API，以跳过API调用退出指令



综上所述，当柜台在同一个交易日内重启时，易达API为投资者提供了三种不同程度的处置方式：

- 重启进程：这种方式最为干净、简单且不易发生错误，策略程序重新从头接收柜台流水并恢复到最新状态。缺点是策略程序会被连带退出，在生产上可能欠缺可操作性。
- 重建API实例：这种方式相对干净且不会导致策略程序退出，重建实例后API会重新推送所有流水。若策略程序保存且需要复用本地流水，可能需要合并API重新下发的流水，而且策略程序需要确保在重建过程中不可以访问被销毁实例的数据，否则会导致程序奔溃，因此需要对策略程序有较为细腻的控制。同时，重建实例过程中也会遇到策略端超时未知单以及外部系统报单的问题需要处理，详情请参考高可用章节内容。
- 重新建立连接：即开启高可用模式下的方式，对策略程序的影响几乎无感知，发生自动切换后仅仅通过notifyEvent(YD_AE_ServerSwitch)告知策略程序发生了高可用切换。考虑到通常策略程序一般都具有处理策略端超时未知单以及外部系统报单的能力，因此本方式对于策略程序最为友好。本模式下具体情况请参考高可用。

3.4.1.1 高可用

为了满足做市商和重视可用性的投资者的需要，易达推出了高可用集群功能，它可以确保最大程度上的业务连续性，当发生硬件宕机、柜台程序错误等故障时，投资者的交易会在最短时间内恢复。

高可用集群由三台服务器组成的热备集群，包括两台运行ydServer的柜台服务器和一台运行ydArb的仲裁服务器。为满足不同用户对于性能和可用性的权衡，主备柜台服务器的硬件配置有多种不同的组合：

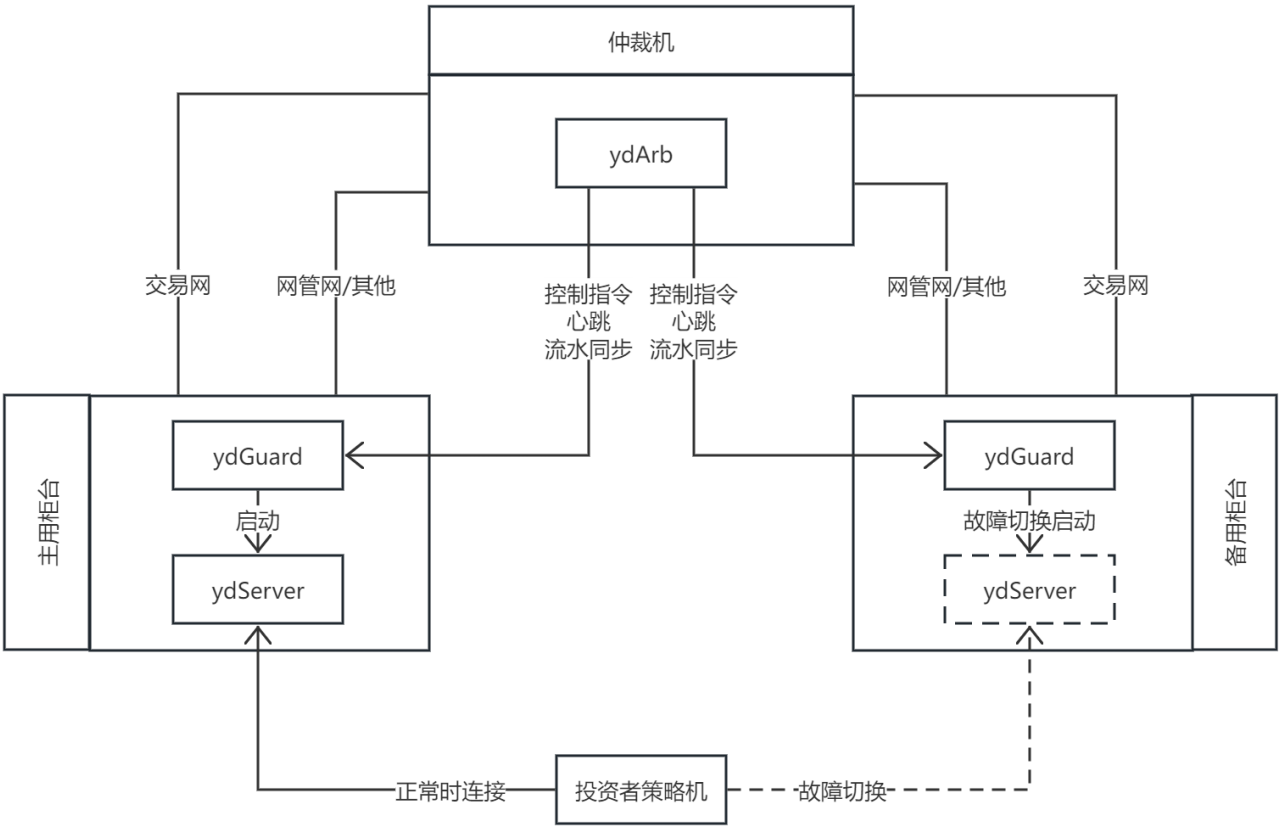
- 最高保护：主备服务器均采用普通服务器，最大限度上确保主用服务器极少发生故障，即便发生故障，备用服务器也可以在极大程度上成功启动并接替交易。在当前版本中，经过性能调优的普通服务器的穿透性能比超频机差500ns，对于做市商、自营等极端关注系统稳定性和交易可用性的用户较为适合。
- 兼顾性能和保护：主用服务器采用超频机，备用服务器采用普通服务器。考虑到目前超频机较为稳定，通常不会发生故障，因此绝大部分时间交易都在超频机上，可以获得最高的性能，一旦超频机发生故障，由于备机使用的普通服务器极为稳定，备用服务器也可以在极大程度上成功启动并接替交易。这种方案兼顾了性能和可用性，当发生切换时放弃一定的性能但是确保了交易的持续性。
- 最高性能：主备服务器均采用超频机，最大程度上保证了交易性能。但是由于备用服务器和主用服务器的上架时间相同，老化时间也相同，因此当发生主备切换时存在一定的可能性备机无法启动，但是目前超频机的稳定性得到了极大的增强，发生问题的概率不大。这种配置组合适合大部分投资者。

高可用集群的三台服务器均应建立仲裁网，用于传输心跳、交易流水和启停控制指令，仲裁网只需保证仲裁机能连通主备服务器即可，主备服务器之间的连通性没有要求。按照每个席位每天约600MB计算，通常的千兆网络就能满足仲裁网的带宽需求，若网管网容量能满足上述带宽要求，可以使用网管网作为仲裁网。为了避免仲裁网网络故障带来的无效切换，建议将投资者的交易网配置为仲裁网的备份。当仲裁网正常工作时，所有流量将在仲裁网上，交易网不会有任何的流量；当仲裁网故障时，才会将流量切换到交易网。若仲裁机发现无法通过仲裁网连通主用服务器，它将通过备用网络尝试联系主用服务器，如果此时仍然无法连接，此时发起主备切换就显得更为合理，因为这个时候投资者也大概率无法连通柜台。

高可用集群的主要组件功能如下：

- ydArb：部署在仲裁机上面的常驻仲裁程序。通过心跳信息监控集群的可用性，当主用节点发生故障时，主动通知备用节点启动接替主用节点。同时还负责将主用节点上的盘中数据（交易所流水和柜台本地流水）持续传输到备用节点，以保证备用节点最快速启动。

- ydGuard：部署在主备用服务器上的ydServer常驻守护进程。检测ydServer的状态并向ydArb通报，同时主用服务器上ydGuard向ydArb传输盘中数据，而备用服务器上ydGuard从ydArb接收盘中数据。主用服务器ydGuard的数据传输相对ydServer的交易而言是异步的，极有可能发生回报已经发送到投资者但是没有同步到备用服务器的情况。



当要连接到高可用集群时，需要按照如下样例修改API的配置文件。

```
RecoverySiteCount=2
TradingServerIP=<ip of master host>
TradingServerPort=<port of master host>
TradingServerIP2=<ip of slave host>
TradingServerPort2=<port of slave host>
```

正常运行时，主用服务器的ydServer进程处于启动状态，备用服务器的ydServer进程处于停止状态，ydArb和ydGuard构成的集群持续将交易流水经由仲裁机实时从主用服务器传送到备用服务器，此时投资者的API连接到主用服务器上面的ydServer交易。

当发生主备切换时，备用服务器上的ydServer会被拉起，并加载同步的盘中数据，尽可能恢复到主用服务器发生故障时的状态。易达API检测到断线后，将轮询主备服务器的IP地址直到重新连上，连上后的后续步骤可以参考交易重连章节内容。由于主备切换固有的复杂性，以下问题是无法避免的，需要投资者自行处理：

- 本地超时未知单：若柜台发生故障的时间点为报单从策略程序发出之后到在柜台发往交易所之前这个时间段内，策略程序会一直等待回报更新报单状态，但是由于报单没有报送到交易所，因此在柜台和交易所的角度来看，这笔报单并不存在，因此也不可能撤单。因此，策略程序需要建立超时机制，如果遇到主备发生切换的时候，应该主动删除策略端超时的本地报单。请注意，本地超时报单与超时未知单发生的原因、产生的后果是不同的，因此处理方式也是不同的。
- 无法查到对应关系的报单和成交：若柜台发生故障的时间点为报单从柜台发出之后到主用服务器的流水同步到备用服务器之前的这个时间段内，显而易见的是发生故障后，策略程序也没有收到对应的交易所报单回报（否则就不存在需要特殊处理的丢单）。在备用服务器完成启动后，易达会保证不会丢失报单本身，但是由于主用服务器的本地流水还没有同步，备用服务器重新发送的这一报单回报是含有OrderSysID、OrderLocalID以及RealConnectionID的，但不会含有OrderRef的。那么投资者在收到这笔回报后，是无法与策略端记录的报单相关联的，当遇到OrderRef为-1这种情况时，

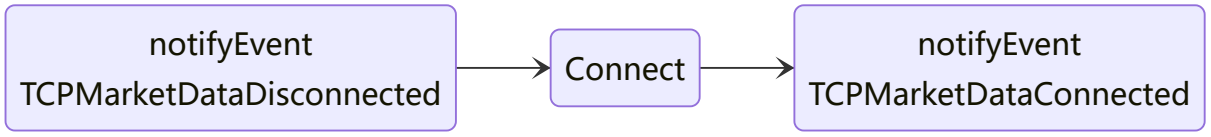
请按照普通的外部报单逻辑一样处理即可，而那笔没有被匹配到的策略端报单则应按照本地超时未知单的处理逻辑进行处理。

单柜台高可用是双柜台高可用版本的特例版本，等同于在双柜台高可用模式下将两个IP地址配置为相同的地址，两者实际效果基本是相同。单柜台高可用模式简化了策略程序与到柜台在同一个交易日内重启时的处理过程，因此推荐投资者尽可能使用单柜台高可用特性。单柜台高可用的对应配置文件的配置样例如下：

```
RecoverySiteCount=1
TradingServerIP=<ip of master host>
TradingServerPort=<port of master host>
```

3.4.2 行情重连

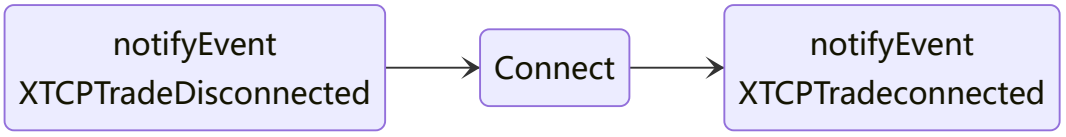
当API探测到与柜台的TCP行情连接中断或者超时后，API会回调notifyEvent(YD_AE_TCPMarketDataDisconnected)通知行情线程断开了与柜台的连接，此后，API会持续发起到柜台的连接，直到重新连接上后API会回调notifyEvent(YD_AE_TCPMarketDataConnected)通知。相比交易重连的过程，行情重连相对简单很多。重连成功后，不需要再次订阅行情，API将会自动重新订阅断线前已订阅合约，并继续收取TCP行情。



3.4.3 XTCP重连

当API探测到与柜台的XTCP交易连接中断或者超时后，API会回调notifyEvent(YD_AE_XTCPTradeDisconnected)通知XTCP线程断开了与柜台的连接。此后，API会持续发起到柜台的连接，直到重新连接上后API会回调notifyEvent(YD_AE_XTCPTradeConnected)通知。

若断线重连期间投资者报单，则API会降级使用普通TCP报单。



3.5 销毁

策略程序可以主动销毁API实例以避免内存泄漏或者避免API主动退出进程，由于易达API的结构较为复杂，无法通过删除实例的方式清理干净，因此易达在ydApi中提供了startDestroy用于实例清理工作，请永远不要尝试直接删除API实例。

```
virtual void startDestroy(void)
```

startDestroy将发起两阶段的异步销毁流程，该函数返回后并不表示清理已经完成。

- 第一阶段的主要工作是尝试关闭所有属于API的连接和线程，完成后将通过YDListener.notifyBeforeApiDestroy通知第一阶段已结束。在此过程中，包括notifyBeforeApiDestroy的回调函数中，所有API中的交易功能都将无法使用，YDListener中的交易回调函数也不会被回调，但是ydApi上所有的查询函数以及ydExtendedApi里管理的数据仍然可以使用。

  ```
  virtual void notifyBeforeApiDestroy(void)
  ```

- 第二阶段的主要工作是销毁所有数据，完成后将通过YDListener.notifyAfterApiDestroy通知清理已全部结束。在此过程中，包括notifyAfterApiDestroy的回调函数中，所有API的功能以及从API中获得的指针均无法使用。

```
virtual void notifyAfterApiDestroy(void)
```

清理结束后，请自行清理您实现并实例化的YDListener子类实例，本函数不会帮您销毁该实例。在策略程序中，您可以重新创建新的API实例。

4 资金

4.1 权益

4.1.1 昨余额

昨余额又被称为日初权益，主要由昨权益组成，期货交易所和证券交易所中昨余额的计算方法并不相同，以下将分别介绍。

> 为了便于经纪商配置使用易达系统，减少不必要的配置错误，易达直接使用现有主席系统的日初文件经过转换后作为易达的日初数据，包括昨余额、日初持仓、保证金率、费率、部分风控规则等。目前易达的期货交易所日初数据可以从CTP的syncmerge或者CTP异构数据接口中直接读取；而证券交易所的股票期权日初数据是从多个数据源获得的，为了满足需求并确保多数据源的勾稽关系，易达引入了一些内部计算。如果经纪商使用的主席柜台是其他品牌的，易达将不断增加新的日初数据转换程序，以满足投资者不受限制的在经纪商分仓参与交易的需求。

昨余额可以在YDAccount.PreBalance中获取。

4.1.1.1 期货交易所昨余额

期货交易所的数据是从CTP的syncmerge或者CTP异构数据接口中读取后简单计算获得的，其计算公式如下。其中昨权益、质押金额、货币质入和质出金额是直接从源数据中直接读取的，固定扣减是在易达系统中设置的。质押金额、货币质入和质出金额默认不计入昨余额，需经纪商主动启用该功能，投资者可查看系统参数中UseCollateral判断是否启用。

$$昨余额 = 昨权益 - 交割保证金 + 质押金额 + 货币质入金额 - 货币质出金额 - 固定扣减_{易达}$$

目前市场上主流的做法是经纪商与投资者签订协议，约定了在每个次席柜台上的固定权益，由经纪商修改日初数据从而达到资金切割的作用，这种做法虽然简单，但是当发生某个交易所市场行情来临时，可能会导致因为资金缺少而没能抓住的行情，即便是可以通过出入金的方式来调度资金，但是人工操作的复杂审核会耽误不少交易机会。因此易达在设计之初就考虑了这种情况，当投资者在多套易达柜台同时交易时，建议经纪商不要拆分投资者的昨权益，使得每套柜台上读取的昨权益都相同，然后通过资金使用限度参数来控制投资者在各套柜台的昨权益占比，结合易达资金调拨系统，在满足同一投资者在所有柜台的资金使用限度加总小于等于100%的前提下，投资者在盘中可以自行修改资金使用限度以实现实时资金调拨的效果。关于资金使用限度的含义请参见今余额。

固定扣减是易达为方便同时在易达和其他次席柜台交易的投资者也能在易达柜台中使用资金使用限度功能，如下表所示，假设某投资者共有100万权益，在易达柜台中分配80万权益，在其他次席柜台中分配了20万权益，那么在使用了固定扣减的情况下，资金使用限度功能仍然能正常工作。

交易所	柜台类型	柜台类型总权益	资金使用限度	占用权益
SHFE	易达	80	40%	32
DCE	易达	80	50%	40
CFFEX	易达	80	10%	8
CZCE	其他	20		20

4.1.1.2 证券交易所昨余额

证券交易所的昨权益可以与期货交易所一样直接读取投资者在沪深两市的昨权益，然后通过资金使用比例进行动态拆分，这是易达推荐的做法。同时为了满足当前市场上投资者的习惯和实际需要，易达也支持按照每个投资者层面定设置的可用资金拆分比例（下称拆分比例，在后台定义，无法在API上读出）拆分，要实现该功能需要从监控中心cfmmc上报文件（只包含沪深两市数据）、中登上海、中登深圳三个数据源中通过计算后获取的，整个计算过程较为复杂，虽然易达无法保证在按照可用资金拆分后易达柜台中显示的可用资金与主席柜台上按照拆分比例计算后的结果相同，但易达一定保证金沪深两市的昨权益相加始终等于主席柜台的昨权益，想了解易达具体计算方法的投资者可以继续阅读，不感兴趣的投资者可以直接跳过以下内容。

在行权交割日由于监控中心cfmmc上报文件（只包含沪深两市数据）、中登上海、中登深圳三份数据的勾稽关系存在差异，因此需要首先计算出适用于易达系统的可用资金作为拆分基础。

$$可用资金_{易达} = 昨权益_{cfmmc} - 沪市保证金_{cfmmc} - 深市保证金_{cfmmc} - 行权分配冻结资金_{中登上海} - 行权分配冻结资金_{中登深圳}$$

然后根据计算得到的可用资金进行拆分就可以得到易达系统中的昨权益。

$$沪市昨权益_{易达} = 沪市拆分比例_{易达} \times 可用资金_{易达} + 沪市保证金_{cfmmc} + 行权分配冻结资金_{中登上海}$$

$$深市昨权益_{易达} = 深市拆分比例_{易达} \times 可用资金_{易达} + 深市保证金_{cfmmc} + 行权分配冻结资金_{中登深圳}$$

简单联立三个公式后能看到，易达系统中的昨权益加总等于监控中心文件中的昨权益。

$$昨权益_{cfmmc} = 沪市昨权益_{易达} + 深市昨权益_{易达}$$

与期货交易所类似，固定扣减功能仍然是生效的，但是如果其他次席柜台系统是按照可用拆分的，那么固定扣减是无法使用的，因此作用不大，通常情况下固定扣减为0。

$$昨余额 = 昨权益 - 固定扣减_{易达}$$

4.1.2 今余额

在易达的设计中，静态权益描述了投资者层面的当日整体权益，该投资者所在任何易达柜台上的静态权益都是相同的，其中入金和出金的金额也是在投资者层面的，即只要在该投资者在主席柜台上出入金，那么该投资者所在任何易达柜台上都会对应入金或者出金，且金额与主席柜台相同。这样就为引入资金使用限度的概念做好了准备。

如果经纪商自行按照固定金额拆分或者按照可用资金比例拆分，那么入金和出金就表达为针对单套柜台的出入金金额，静态权益也仅仅表示该投资者在对应柜台中的当日权益，而且在这种情况下不应该使用资金使用限度，即资金使用限度保持为100%，否则会造成资金错乱。

使用ydExtendedApi的投资者可以调用YDExtendedAccount.staticCashBalance()获取静态权益。出入金净额的计算方法请参考出入金净额。

$$静态权益 = 昨余额 + 出入金净额$$

无论在何种使用模式中，今余额都表示当前投资者在对应柜台中的当日权益。如果按照易达原始设计来使用柜台，那么就可以使用资金使用限度来完成自动出入金同步以及投资者自主的资金调拨功能。

使用ydExtendedApi的投资者可以在YDExtendedAccount.Balance获取今余额。

$$今余额 = (静态权益 - 冻结出金) \times 资金使用限度$$

4.1.2.1 资金使用限度

易达支持通过设置资金使用限度分隔多套系统的日初权益，只要每套柜台的资金使用限度加总不超过100%，就可以保证金不会透支。资金使用限度的设置值可以从YDAccount.MaxMoneyUsage中获取。

在经纪商没有拆分资金且合理设置多套易达柜台的资金使用限度的条件下，投资者可以实现自动出入金和自主资金调拨的功能，下面将通过一个例子来说明。

例如，某投资者同时在中金所、大商所、上期所三套易达系统中交易，其静态权益为100万元，资金使用限度分别为50%、30%和20%。为求简单，例子中忽略了保证金的影响，在任何情况下若出金和设置资金使用限度会导致可用资金不足的都会被系统拦截。因此，默认以下操作都不触发该限制。

	中金所	大商所	上期所
初始资金使用限度	50%	30%	20%
今余额	50	30	20
入金100万后的今余额	100	60	40
上期所需紧急调拨资金，设置后的资金使用限度	10%	20%	70%
调拨后今余额	20	40	140
出金50万后的今余额	15	30	105

资金同步系统ydSync可以即时读取CTP风控系统的出入金流水然后同步到所有易达柜台；资金调拨系统ydFundManager实现了多套易达柜台的资金调拨功能，经纪商可以给投资者开立调拨账户以便投资者登录资金调拨系统自行调拨资金。如有需要请联系经纪商安装，但是上述功能的前提条件是经纪商没有做资金拆分。

4.1.3 其他权益

为了最终计算资金权益和市值权益，易达把资金权益和市值权益公式中与交易有关的部分称为动态权益。动态权益只与柜台所在交易所有关，两套部署在相同交易所的拥有管理席位的柜台对于同一投资者的动态权益是相同的，但是若上述的两套中的一套配置的是全非管理席位的，这套全非管理席位的柜台无法收到其他柜台的成交数据，此时只有那套拥有管理席位的柜台的动态权益是准确的。当然，如果该交易所只有一套柜台，那么无论该使用的是何种席位，其动态权益总是准确的。单套柜台的动态权益的计算公式如下所示：

$$动态权益 = 平仓盈亏 + 期货持仓盈亏 + 现金收支 - 手续费$$

在动态权益概念的帮助下，单套柜台的资金权益和市值权益的计算公式如下所示，基于上述原因，单套柜台的资金权益和市值权益也同样具有简单可加性：

$$今资金权益 = 静态权益 \times 资金使用限度 + 动态权益$$

$$昨资金权益 = 昨余额 \times 资金使用限度$$

$$今市值权益 = 今资金权益 + 今期权市值$$

$$昨市值权益 = 昨资金权益 + 昨期权市值$$

上述权益只能在ydExtendedApi中获取：

- 动态权益：调用YDExtendedAccount.dynamicCashBalance()获得
- 今资金权益：调用YDExtendedAccount.cashBalance()获得
- 昨资金权益：调用YDExtendedAccount.preCashBalance()获得
- 今市值权益：调用YDExtendedAccount.marketValue()获得
- 昨市值权益：调用YDExtendedAccount.preMarketValue()获得

4.2 可用资金

$$可用资金 = 今余额 + 平仓盈亏 + min(期货持仓盈亏, 0) + 现金收支 - 手续费 \\ -期货保证金 - 卖出期权保证金 - 期权行权保证金 - 行权分配冻结金额$$

使用ydExtendedApi的投资者可以调用YDExtendedAccount.usable()获取可用资金。监管规定持仓亏损要扣除，因此易达柜台在计算可用资金时扣除了持仓亏损。期货法颁布后，删除了持仓盈利不计入可用资金的要求，各地证监局对此也有不同的规定，因此，从 `1.386.40.38` 版本开始，经纪商可以为客户设置将持仓盈利计入可用资金的功能开关，打开后该投资者的持仓盈利将计入可用资金。投资者可以通过查看YDAccount中AccountFlag是否设置了YD_AF_UsePositionProfit来判断持仓盈利是否计入可用资金。

> YDExtendedAccount.Available与可用资金比较相像，从下列公式可以看出Available并没有包括持仓亏损，因此Available是不能等同于可用资金来使用的。
>
> $$Available = 今余额 + 平仓盈亏 + 现金收支 - 手续费 \\ -期货保证金 - 卖出期权保证金 - 期权行权保证金 - 行权分配冻结金额$$

由于API与柜台刷新的时机以及刷新时使用的价格并不完全相同，可能会导致在API上显示资金足够的但是被柜台因资金不足而拒绝的情况发生。一般这类问题会发生在资金利用率较高的情况下，如果您希望确认因资金不足而被柜台拦截的单子的具体情况，可以联系经纪商从柜台日志中获取拒单时的资金情况，请注意其中Available并不是可用资金，需要扣除PositionProfit的亏损部分才行。

可用资金和Available需要刷新重算后才能显示根据最新价计算的结果，具体刷新方式请参见资金刷新机制。

4.3 出入金净额

出入金净额的计算公式为：

$$出入金净额 = 累计入金金额 - 累计出金金额$$

当投资者有出入金发生时，可以通过YDAccount.Deposit获得累计入金金额，通过YDAccount.Withdraw获得累计出金金额，YDAccount.Deposit和YDAccount.Withdraw在一个交易日内是单调递增的。

冻结出金金额可以通过YDAccount.FrozenWithdraw获得，冻结出金金额会随着经纪商执行冻结出金操作而不断累加，经纪商解除冻结时YDAccount.FrozenWithdraw会相应减少，或者经纪商正式出金时也会相应扣减被与出金金额相同的冻结金额。

当发生出入金或者出金冻结时，会通过notifyAccount通知到投资者，由于YDAccount其他字段发生变化时也会通知到投资者，因此投资者需要在程序中检测上述三个字段以确认是否发生了出入金。

```
virtual void notifyAccount(const YDAccount *pAccount)
```

4.4 现金收支

现金收支可以从YDExtendedAccount.CashIn读取，期权权利金和现货收支均计入现金收支。

期权和现货买方报单时冻结现金收支，随着报单逐步成交，现金收支中的报单冻结资金会被逐步释放，并相应增加现金收支中的成交资金扣减，若报单最终撤单，则一次性释放报单冻结资金。

期权和现货卖方报单时不改变现金收支，随着报单逐步成交，现金收支中逐步增加成交资金，若报单最终撤单，则不改变现金收支。

4.4.1 现货报单收支

现货买方报单收支的计算公式如下所示，当报单为市价单时，报单价格为涨停板价，当报单为限价单/FAK/FOK时，报单价格为报单价格。债券价格只有当交易证券为债券时才有值，其他现货为0：

$$现货买方报单收支 = -(报单价格 + 债券利息) \times 报单量$$

现货卖方报单收支始终为0。

4.4.2 现货成交收支

现货买方成交收支的计算公式如下所示：

$$现货买方成交收支 = -(成交价 + 债券利息) \times 成交量$$

现货卖方成交收支的计算公式如下所示：

$$现货卖方成交收支 = (成交价 + 债券利息) \times 成交量$$

债券价格只有当交易证券为债券时才有值，其他现货为0。

4.4.3 期权报单权利金

期权买方报单权利金的计算公式为：

$$期权买方报单权利金 = -报单价格 \times 合约乘数 \times 报数量$$

其中，当报单为市价单时，报单价格为涨停板价，当报单为限价单/FAK/FOK时，报单价格为报单价格。

期权卖方报单权利金的计算公式为：

$$期权卖方报单权利金 = 报单价格 \times 合约乘数 \times 报数量$$

其中，期权卖方报单价格受到YDSystemParam中（Name, Target）为（SellOrderPremiumBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价
- YD_CBT_OrderPrice：当报单为市价单时，价格为跌停板价，当报单为限价单/FAK/FOK时，价格为报单价格

- YD_CBT_None：不增加权利金，等同于价格为0，即期权卖方在报单时不会获得权利金。这是默认配置的方式，也是主流系统当前使用的方式。

4.4.4 期权成交权利金

期权买方成交权利金的计算公式为：

$$期权买方成交权利金 = -成交价 \times 合约乘数 \times 成交量$$

期权卖方成交权利金的计算公式为：

$$期权卖方成交权利金 = 成交价 \times 合约乘数 \times 成交量$$

4.5 期货持仓盈亏

$$期货多头持仓盈亏 = ((最新价 - 开仓价) \times 今持仓量 + (最新价 - 昨结算价) \times 昨持仓量) \times 合约乘数$$

$$期货空头持仓盈亏 = ((开仓价 - 最新价) \times 今持仓量 + (昨结算价 - 最新价) \times 昨持仓量) \times 合约乘数$$

持仓盈亏的刷新逻辑请参见资金刷新机制。

4.6 平仓盈亏

$$多头平今仓盈亏 = (平仓价–开仓价) \times 合约乘数 \times 平仓手数$$

$$多头平昨仓盈亏 = (平仓价–昨结算) \times 合约乘数 \times 平仓手数$$

$$空头平今仓盈亏 = (开仓价–平仓价) \times 合约乘数 \times 平仓手数$$

$$空头平昨仓盈亏 = (昨结算–平仓价) \times 合约乘数 \times 平仓手数$$

平仓配对持仓明细的逻辑请参见衍生品持仓模型。

4.7 期权市值

$$昨期权市值 = 期权买持仓量 \times 昨结算价 \times 合约乘数 - 期权卖持仓量 \times 昨结算价 \times 合约乘数$$

$$今期权市值 = 期权买持仓量 \times 最新价 \times 合约乘数 - 期权卖持仓量 \times 最新价 \times 合约乘数$$

昨期权市值和今期权市值分别可以通过YDExtendedAccount.PrePositionMarketValue和YDExtendedAccount.PositionMarketValue获取。

今期权市值需要刷新后才能显示根据最新价计算的结果，具体刷新方式请参见资金刷新机制。

4.8 手续费

使用ydExtendedApi的投资者可以通过YDExtendedAccount.Commission获取该账户的总手续费。

4.8.1 现货手续费

报单手续费在报单成功后收取，撤单手续费在撤单成功后收取，成交手续费在成交的时候收取。上交所、深交所的现货买卖报单发出后预先冻结最大可能成交手续费，待委托全部成交或者撤单后，再相应补回差额。

$$印花税 = min(max(成交量 \times (成交价 \times 以金额计算的印花税率 + 以数量计算的印花税率), 最小印花税金额), 最大印花税金额)$$

$$证管费 = min(max(成交量 \times (成交价 \times 以金额计算的证管费率 + 以数量计算的证管费率), 最小证管费金额), 最大证管费金额)$$

$$经手费 = min(max(成交量 \times (成交价 \times 以金额计算的经手费率 + 以数量计算的经手费率), 最小经手费金额), 最大经手费金额)$$

$$过户费 = min(max(成交量 \times (成交价 \times 以金额计算的过户费率 + 以数量计算的过户费率), 最小过户费金额), 最大过户费金额)$$

$$佣金 = min(max(成交量 \times (成交价 \times 以金额计算的佣金率 + 以数量计算的佣金率), 最小佣金金额), 最大佣金金额)$$

手续费率的获取方式参见现货交易规费率和现货佣金费率。

4.8.2 衍生品交易手续费

报单手续费在报单成功后收取，撤单手续费在撤单成功后收取，成交手续费在成交的时候收取。上交所、深交所的股票期权买开报单以及其他期货交易所开仓报单发出后预先冻结成交手续费，冻结成交手续费的计算公式同开仓成交手续费，其中开仓量使用报单量，成交价格则根据报单的价格类型有所不同，市价单使用涨停板价格，非市价单使用报单价。平仓报单不冻结手续费。

$$报单手续费 = 报单笔数 \times 报单每笔手续费$$

$$撤单手续费 = 撤单笔数 \times 撤单每笔手续费$$

$$开仓成交手续费 = 开仓量 \times (开仓成交手续费率 \times 成交价格 \times 合约乘数 + 开仓成交每手手续费)$$

$$平今仓成交手续费 = 平今仓量 \times (平今仓成交手续费率 \times 成交价格 \times 合约乘数 + 平今仓成交每手手续费)$$

$$平昨仓成交手续费 = 平昨仓量 \times (平昨仓成交手续费率 \times 成交价格 \times 合约乘数 + 平昨仓成交每手手续费)$$

$$手续费 = \sum_{所有报单} 报单手续费 + \sum_{所有撤单} 撤单手续费$$

$$+ \sum_{所有开仓成交} 开仓成交手续费 + \sum_{所有平今成交} 平今仓成交手续费 + \sum_{所有平昨成交} 平昨仓成交手续费$$

手续费率的获取方式参见手续费率。

不同交易所平仓手续费的计算方法有所区别，易达在YDExtendedPosition.YDPositionForCommission上维护了该持仓用于手续费计算的昨仓数量，投资者可以通过计算YDExtendedPosition.Position-YDExtendedPosition.YDPositionForCommission获得用于手续费计算的平今仓数量。

为方便投资者分辨对应合约优先平今还是平昨，投资者可以通过YDExchange.CloseTodayFirst获得该合约对应交易所是否优先平今。

下面简要说明各个交易所在计算平仓手续费时，使用的今仓和昨仓持仓明细的顺序：

> 请注意用于手续费计算的持仓明细、用于计算平仓盈亏的持仓明细以及用于计算组合保证金的持仓明细都是单独维护的，因此请根据实际使用场景选择正确的持仓信息

- 对于上期所和能源所，根据平今或者平昨指令决定
- 对于中金所、大商所、广期所和郑商所，优先平今仓
- 对于上交所和深交所，优先平昨仓

上交所卖开成交不收取手续费，与手续费参数设置无关。

4.8.3 期权行权手续费

期权行权手续费的计算公式为：

$$期权行权手续费 = 行权量 \times (行权每手手续费 + 行权金额手续费率 \times 行权价 \times 期权合约乘数)$$

行权手续费率的获取方式参见手续费率。

4.8.4 衍生品信息量申报费

目前上期所、能源所所有产品，大商所、郑商所部分产品开始按照信息量收取申报费，易达支持按照OTR和信息量分级分档收取信息量申报费，具体收费标准请参考各交易所相关通告。

由于在某些情况下可减免信息量申报费，例如除上期所、能源所之外的交易所的询价单不计入信息量，做市商不收取信息量申报费等，投资者可以检查YDAccountInstrumentInfo.ExemptMessageCommissionYDOrderFlag与报单的YDOrderFlag判断该报单是否计入信息量，若YDOrderFlag>YDAccountInstrumentInfo.ExemptMessageCommissionYDOrderFlag则不计入，否则需计入。

在与监管公式原则上保持一致的前提下，为保证在极端条件下OTR的合理性，易达使用如下OTR的计算公式：

$$OTR = \frac{max(\text{信息量}, 1)}{max(\text{有成交的报单数}, 1)} - 1$$

上述公式中，有成交的报单的定义为：若某笔报单部分或全部成交，则该笔报单计为1笔有成交的报单，1笔报单若分多次成交不重复统计。

信息量计算方式如下：

- 限价单：若全部成交仅计1笔报单笔数；若撤单，则计1笔报单笔数和1笔撤单笔数
- FAK/FOK定单：若全部成交仅计1笔报单笔数；若未成交或未全部成交而产生撤单，则计1笔报单笔数和1笔撤单笔数
- 市价单：若全部成交仅计1笔报单笔数；若未成交或未全部成交而产生撤单，则计1笔报单笔数和1笔撤单笔数
- 询价单，上期所、能源所、郑商所、广期所的期权询价单增加1笔信息量，上期所、能源所、郑商所、广期所的期货询价单和其他交易所询价单不计
- 组合（套利）定单：组合定单的各腿合约信息量分别计入各腿合约上
- 交易所强行平仓订单，非期货公司强平订单：均计入信息量
- 强制减仓定单：均不计入信息量
- 限价单、FAK单、FOK单、市价单的错误报单，均不计入信息量
- 限价单的错误撤单，均不计入信息量
- 组合和解组指令、期权行权和放弃行权、期权对冲、履约后对冲的报单、撤单、错误报单、错误撤单，均不计入信息量
- 做市商在做市品种上免收信息量申报费
- 广期所期权的信息量申报费基于期权系列汇总计算，同一产品、同一到期日的期权合约加总后按照信息量梯度收费。期货和其他交易所的期权均在合约上汇总计算

报单时，易达将按照该笔报单的最坏可能情况收取申报费，待收到回报后依照回报的实际情况扣回多计信息量。例如柜台收到限价单后计入2信息量，若该报单被拒绝，则扣回2信息量，若该报单后续撤回，则不扣回信息量。其中，询价单信息量可以通过YDAccountInstrumentInfo.RFQCount获取。由于询价单不对普通投资者发送回报，在上期所和能源所下期权询价单时，如果询价单产生的信息量申报费没有通过，该询价单会被直接抛弃，而不会通知客户，但该错误信息会记录在柜台inputFlow.txt中，这和其他询价单失败的情况相同。

信息量申报费采用分段累计的计算方法，可参考信息量申报费率中的计算实例。使用ydExtendedApi的投资者可以通过YDExtendedAccount.MessageCommission获取该账户的总信息量申报费。

由于投资者程序的不当控制，可能会造成收取大量意料外信息量申报费的情况。为了规避信息量申报费对投资者造成的重大损失，请联系经纪商设置事前信息量风控或者事后信息量风控。

4.9 资金刷新机制

易达保证金、持仓盈亏和持仓市值并不是从柜台查询而来，而是直接在客户端使用与柜台相同的方法计算得来，因此，需要在盘中实时刷新保证金和持仓盈亏，以使得可用资金与柜台实际保持一致。

自从`1.98`起易达API提供了多种盘中自动刷新保证金率和持仓盈亏的机制，对于使用不同API的投资者给予了不同程度的支持。通过设置API配置文件中的RecalcMode参数可以指定API的三种刷新机制，分别是关闭、只订阅行情和自动三种模式，下面将分别介绍这三种机制。

4.9.1 关闭模式

关闭即完全关闭自动刷新机制，需要由投资者自行决定刷新的时机和方式。

对于使用ydApi的投资者，由于ydApi并没有计算资金和持仓，因此，所有的刷新工作就完全由投资者自行进行，API完全不参与整个过程。

对于使用ydExtendedApi的投资者，当投资者需要刷新时，可以通过调用recalcMarginAndPositionProfit刷新保证金和持仓盈亏，调用recalcPositionMarketValue刷新持仓市值，这两个函数均无输入参数。由于关闭模式不会自动订阅行情，请务必保证接收TCP行情（ConnectTCPMarketData=yes），否则将始终按照日初行情计算刷新。

4.9.2 订阅行情模式

订阅行情模式可以帮助投资者自动订阅并刷新当日所有交易过的合约或者持仓合约的行情，如果自动订阅了一个期权合约，就会自动订阅其基础合约，以便准确计算期权保证金。但是自动订阅的行情并不会直接通过notifyMarketData发送给投资者，如果需要通过notifyMarketData收取的，请使用subscribe单独订阅。本模式对使用ydApi和ydExtendedApi的投资者均有帮助。

对于使用ydApi的投资者，行情到达时会刷新API中YDMarketData行情，投资者可以直接读取对应合约的价格来自行刷新。

对于使用ydExtendedApi的投资者，行情会保存在ydExtendedApi之中，当投资者需要刷新时，可以通过调用recalcMarginAndPositionProfit刷新保证金和持仓盈亏，调用recalcPositionMarketValue刷新持仓市值，这两个函数均无输入参数。

4.9.3 自动模式

自动模式包含了订阅行情模式的所有功能，并在此基础上，针对不同类型的API提供了更进一步的支持。

对于使用ydApi的投资者，为了能帮助投资者错开可能的报单时间，API会通过notifyRecalcTime通知投资者系统检测到的安全刷新时间，投资者可以在这个函数中调用自己的刷新方法。

安全刷新时间是依据上一次收到的TCP行情时间，并结合API配置文件中的RecalcMarginPositionProfitGap和RecalcFreeGap的设置值计算出来的，其中RecalcMarginPositionProfitGap是指连续两次刷新的最小间隔，最小可以被设置为1000毫秒，低于1000毫秒的设置将会被调整为1000毫秒，RecalcFreeGap是指安全刷新时间与行情到达前后的安全距离，其设置区间为0至100毫秒，所有超出该范围的设置值都会被调整为距离该值最近的合法值。

具体计算方法为，当距离上次安全刷新时间达到RecalcMarginPositionProfitGap之后，以API上次收到的TCP行情为0毫秒，以250毫秒长度为一个检测段，在该检测段内，去除该检测段前后RecalcFreeGap毫秒后剩余的时间段的开始时间即为安全刷新时间。假设RecalcFreeGap为100毫秒，收到行情的时点是第0毫秒，在该250毫秒时间段内扣除前后100毫秒后剩余的时间段为第100毫秒至第150毫秒，长度为50毫秒的时间段，而安全刷新时间即为第100毫秒。

考虑到行情会持续到达，导致安全刷新时间可能会被持续错过，因此易达设置了一个保护时间，即距离上次刷新时间超过RecalcMarginPositionProfitGap的3倍时间还没有刷新的话，ydApi则会通过notifyRecalcTime通知客户强制刷新。

对于使用ydExtendedApi的投资者，API会在安全刷新时间自动调用recalcMarginAndPositionProfit和recalcPositionMarketValue刷新，无需投资者编写任何代码，此时，notifyRecalcTime会在调用完成上述两个刷新函数后通知到投资者，投资者可以自行决定是否需要在该时点做除了刷新之外的其他工作。

易达已经尽可能帮助投资者避开可能的报单时间，但是有可能投资者的报单行为超出了易达的设计预期，因此，如果您发现使用自动模式的刷新时间与报单时间冲突的，可以考虑调整参数或者使用订阅行情模式，自行寻找刷新时间。

5 持仓

5.1 衍生品持仓模型

各个交易所管理持仓的方式是不同的，有的交易所严格区分了今仓和昨仓并有专门平今和平昨指令；有的交易所并不严格区分今仓和昨仓，也没有专门的平今或者平昨指令，只是在个别业务上（例如平今仓手续费）有区分，因此，易达根据交易所的持仓模型实现了相应的持仓模型。为了便于投资者区分当前交易所使用的持仓模型，投资者可以查看YDExchange.UseTodayPosition的值：当该值为True时则表示该交易所区分了今仓和昨仓，易达系统将会用两条持仓来对应今仓和昨仓；当该值为False时则表示该交易所不区分今仓和昨仓，易达系统只用一条持仓来表示对应的持仓。

YDExtendedApi的持仓组织方式与易达柜台内部的组织方式基本一致，因此下面将以YDExtendedApi的持仓YDExtendedPosition为例，介绍在上述两种持仓模型中用于计算手续费、平仓盈亏和组合业务的规则和相关逻辑。YDExtendedPosition的与持仓模型有关的字段如下所示，主键为合约、持仓日期、持仓方向和投保标志。

字段	说明
getInstrument()	合约
PositionDate	持仓日期 YD_PSD_History：昨仓 YD_PSD_Today：今仓 若交易所不区分今仓和昨仓，一律使用YD_PSD_History
PositionDirection	持仓方向 YD_PD_Long：多头持仓 YD_PD_Short：空头持仓
HedgeFlag	投保标志，期货交易所与股票期权交易所的定义不同。 期货交易所如下： YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。为便于编写程序，易达为是否支持套利交易和持仓提供了参数化表示，YDExchange.UseArbitragePosition为true时表示该交易所支持套利交易和持仓，否则为不支持 YD_HF_Hedge=3：套保 股票期权交易所如下： YD_HF_Normal=1：普通 YD_HF_Covered=3：备兑
Position	当前主键下的总持仓量 若区分今仓和昨仓，则分别表示合约在某持仓方向、某投保标志上的今仓和昨仓的持仓量 若不区分今仓和昨仓，则表示合约在某持仓方向、某投保标志上的总持仓量
PositionDetailList	当前主键下的用于盈亏计算的持仓链表，是由尚未平仓的成交按照成交先后组成的链表 平仓时总是从链表头上开始逐一配对平仓，即先开先平原则
YDPositionForCommission	当前主键下的用于手续费计算的昨持仓量，仅在不区分今仓和昨仓时有效，此时用于手续费计算的今持仓量为Position-YDPositionForCommission

传统组合持仓是已经参与组合的持仓链表，按照组合的先后顺序依次排序；单腿持仓是尚未参与组合的持仓链表，按照成交先后顺序依次排序。由于传统组合持仓明细的结构变化较为频繁，易达并没有在上述持仓模型中直接开放传统组合持仓和单腿持仓的入口，如果需要明细请参考相关查询函数。目前只有不区分今仓和昨仓的交易所支持组合和解组。

- 当交易所区分今仓和昨仓时，
    - 开仓时会对PositionDate为YD_PSD_Today的持仓完成以下操作：

        - 增加对应持仓的Position以更新总持仓数量
        - 将新开仓成交添加到上述持仓的用于盈亏计算的持仓链表的队尾
    - 平仓时投资者需要指定YDInputOrder.OffsetFlag为YD_OF_CloseToday或者YD_OF_CloseYesterday分别表示平今或者平昨（如果设置为YD_OF_Close则会被自动转换为YD_OF_CloseYesterday），会对受影响的持仓完成以下操作：
        - 扣减今仓或者昨仓的Position以更新今仓和昨仓的数量
        - 按照先开先平的顺序更新盈亏计算的持仓链表
- 当交易所不区分今仓和昨仓时，
    - 开仓时会对PositionDate为YD_PSD_Yesterday的持仓完成以下操作：
        - 增加对应持仓的Position以更新总持仓数量
        - 将新开仓成交添加到上述持仓的用于盈亏计算的持仓链表的队尾
        - 若交易所支持传统组合持仓业务，将新开仓成交添加到上述持仓的传统组合持仓链表的队尾
    - 平仓时投资者需要指定YDInputOrder.OffsetFlag为YD_OF_Close表示平仓（如果设置为YD_OF_CloseToday或者YD_OF_CloseYesterday则会被自动转换为YD_OF_Close），会对PositionDate为YD_PSD_Yesterday的持仓完成以下操作：
        - 扣减对应持仓的Position以更新总持仓数量
        - 按照先开先平的顺序更新盈亏计算的持仓链表
        - 若YDExchange.CloseTodayFirst为false，则优先扣减YDPositionForCommission；若YDExchange.CloseTodayFirst为true，只有当“平仓量>用于手续费计算的今持仓量”时，才需要从YDPositionForCommission中扣减用于手续费计算的今持仓量无法覆盖的部分
        - 若交易所支持传统组合持仓自动解组合业务，则优先按照先开先平的顺序匹配单腿持仓中的项直到覆盖平仓量；若无法完全覆盖的，则继续按照先开先平的顺序匹配传统组合持仓中的项直到覆盖剩余平仓量，并且修改涉及到的传统组合持仓中对方腿的传统组合持仓以及单腿持仓
    - 组合时，从待组合的两个持仓的单腿持仓的头部开始取出能覆盖组合数量的项，并添加到两个持仓的传统组合持仓的末尾；解组合时，从待组合的两个持仓的传统组合持仓链表里取出能覆盖解组合数量的项，同时插入到两个持仓的单腿持仓的合适位置，即能保持单腿持仓中的项按照成交先后顺序排序

除了上述与持仓模型有关的字段外，YDExtendedPosition额外提供了以下扩展持仓字段。

字段	说明
getYDPosition()	昨持仓。以先开先平原则计算所得的昨持仓量
PositionByOrder	委托持仓量。根据委托回报中的成交量计算得到的持仓量。可平仓量为 $min(Position, PositionByOrder) - CloseFrozen$
PossibleOpenVolume	潜在开仓量。持仓对应的未完成开仓委托的委托量之和，委托完成后（全成、撤单、错单）潜在开仓量将扣减委托中的未成交量
OpenFrozen	开仓冻结量。持仓对应的未完成开仓委托的未成交量之和，委托完成后（全成、撤单、错单）该委托的开仓冻结量变为0
CloseFrozen	平仓冻结量，被平仓冻结的持仓无法平仓或者申请行权、放弃行权。持仓对应的未完成平仓委托的未成交量，行权冻结量、放弃行权冻结量以及沪深两市股票期权传统组合持仓之和，委托完成后（全成、撤单、错单、解组合）该委托的平仓冻结量变为0
ExecFrozen	行权冻结量。持仓对应的行权申请的委托量之和
AbandonExecFrozen	放弃行权冻结量。持仓对应的放弃行权申请的委托量之和
TotalCombPositions	参与组合的持仓量
CombPositionCount	该持仓参与的传统组合持仓明细的记录数

字段	说明
TotalOpenPrice	开仓均价总成本。 开仓时，增加开仓成交的开仓成本，开仓成本 = 开仓价 × 开仓量。 平仓时，以先开先平的顺序，减去对应的开仓成交的开仓成本，当涉及多个开仓成交时，累计扣除涉及的开仓成交的开仓成本
getOpenPrice()	开仓均价。开仓均价 = 开仓均价总成本 ÷ 持仓量，持仓量为0时开仓均价为0
TotalOriginalOpenPrice	持仓均价总成本。 开仓时，增加开仓成交的开仓成本，开仓成本 = 开仓价 × 开仓量。 平仓时，减去以持仓均价计算的成本，即持仓均价 × 平仓量，保持持仓均价不变
getOriginalOpenPrice()	持仓均价。持仓均价 = 持仓均价总成本 ÷ 持仓量，持仓量为0时持仓均价为0
PositionProfit	期货持仓盈亏。详情请参考期货持仓盈亏。
CloseProfit	逐日盯市合约的平仓盈亏。大商所RULE保证金中，期权合约为逐日盯市合约，计算在本字段中，其他情况下期权平仓盈亏计算在非逐日盯市平仓盈亏中。详情请参考平仓盈亏。
OtherCloseProfit	非逐日盯市合约的平仓盈亏。详情请参考平仓盈亏。
Margin	以设定价格计算得到的保证金，当启用新组合保证金业务后始终为0。设定价格请参考YDSystemParam。
MarginPerLot	以昨结算价计算的每手保证金

5.2 现货持仓模型

现货交易中，不同的交易业务会产生或使用不同的持仓类型，目前易达支持历史、今日竞价买入和今日申赎三种类型持仓。

字段	说明
m_pAccountInstrumentInfo	指向账户合约信息的指针
HoldingPiece[YD_CHT_History]	历史仓持仓，字段参考YDHoldingPiece结构
HoldingPiece[YD_CHT_TodayTrading]	今日竞价买入持仓，字段参考YDHoldingPiece结构
HoldingPiece[YD_CHT_TodayCreationRedemption]	今日申赎持仓，字段参考YDHoldingPiece结构
TotalHolding	持仓汇总，字段参考YDHoldingPiece结构
ExternalSellFrozen	外部冻结量，包括司法冻结、股票期权接管冻结
HoldingAdjustStatus	外部系统转入转出持仓状态，这部分持仓已被实际计入各种类型实际持仓

YDHoldingPiece结构的说明如下表：

字段和方法	说明
Holding	持仓量余额
BuyFrozen	买入冻结数量
SellFrozen	卖出冻结数量
TotalCost	买入成本，卖出不影响

字段和方法	说明
BuyFrozenCost	买入冻结成本
double AverageCost(int multiple=1)	开仓均价，计算公式：TotalCost/Holding
int MaxPossibleHolding()	最大可能持仓量，计算公式：Holding+BuyFrozen
int MaxPossibleSellVolume()	最大可能卖出持仓量，计算公式：Holding-SellFrozen
double MaxPossibleCost()	最大可能成本，计算公式：TotalCost+BuyFrozenCost

YDHoldingAdjustStatus结构的说明如下表：

字段	说明
TotalTransferInVolume	外部系统转入持仓量
TotalTransferOutVolume	外部系统转出持仓量
TotalTransferInPrice	外部系统转入持仓成本
TotalTransferOutPrice	外部系统转出持仓成本

由于竞价卖出时，可用持仓量的计算较为复杂，易达在YDExtendedApi中提供了getMaxSellVolume函数，用于获取当前竞价可卖数量。

```
virtual int getMaxSellVolume(const YDExtendedHolding *pHolding)
```

5.3 股票期权现货持仓模型

股票期权现货持仓只会包含属于沪深交易所股票期权标的物的证券，即ETF和待沪深交易所推出个股期权后的股票。股票期权现货持仓仅用于股票期权业务，与股票期权业务中的备兑和行权业务配套使用。即便将股票期权和现货业务部署在同一套柜台上，对于相同的证券，股票期权现货持仓也完全独立于现货持仓。

YDExtendedApi的持仓组织方式与易达柜台内部的组织方式基本一致，因此下面将以YDExtendedApi的持仓YDExtendedSpotPosition为例，介绍期权现货持仓的相关概念。YDExtendedSpotPosition的与持仓模型有关的字段如下所示，主键为投资者和证券。

字段	说明
m_pAccountInstrumentInfo	指向账户合约信息的指针
Position	股票期权现货持仓量
ExchangeFrozenVolume	用于备兑开仓的交易所锁定量
CoveredVolume	备兑开仓占用量
ExecVolume	E日看跌期权行权冻结量
ExecAllocatedVolume	净行权分配量，始终等于YDSpotPrePosition中数量，盘中不变
ExecAllocatedAmount	净行权分配金额，始终等于YDSpotPrePosition中数量，盘中不变
ExecAllocatedFrozenVolume	行权分配冻结量，始终等于YDSpotPrePosition中数量，盘中不变
ExecAllocatedFrozenAmount	行权分配冻结金额，始终等于YDSpotPrePosition中数量，盘中不变

与其他持仓模型不同，股票期权现货持仓的日初值和盘中实时值与采用的业务模型有关，易达支持增量和全量两种业务模型，增量模型相较全量模型更符合股票期权业务本质，更具通用性，因此易达将在合适的时间将全量模型迁移至增量模型。目前，期货公司部署的易达股票期权柜台使用全量模型，证券公司部署的则使用增量模型。

股票期权现货持仓业务模型是指日初和盘中股票期权柜台与现货柜台、沪深交易所交易系统和中登公司结算系统之间持仓和金额的关系。当采用不同的业务模式时，资金持仓的含义是不同的，投资者在交易时也需要使用不同的API接口。股票期权现货持仓量、行权分配冻结量、行权分配冻结金额与模型有关，其计算方法在具体模型中予以说明；交易所锁定量、备兑量、行权冻结量、净行权分配量、净行权分配金额与模型无关，说明如下：

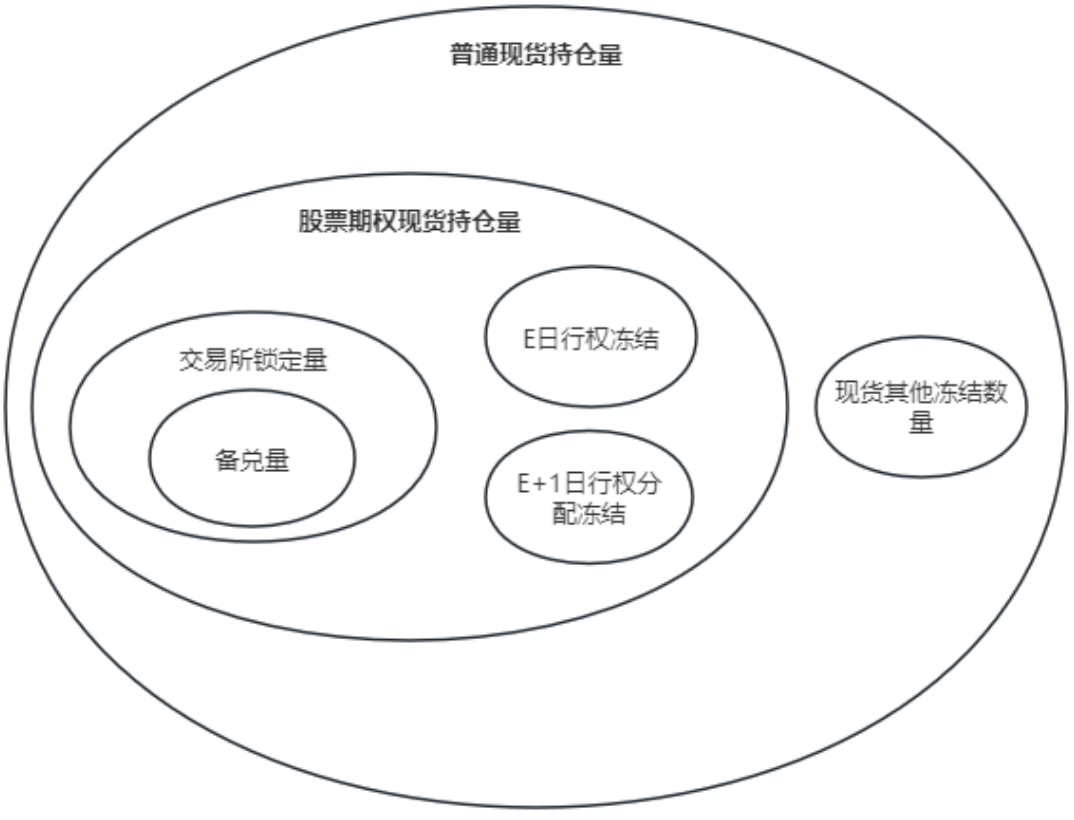
- 备兑量CoveredVolume根据期权备兑持仓计算而得，始终使用交易所合约乘数计算
- 交易所锁定量ExchangeFrozenVolume通常日初时等于备兑量，但当除权除息日或者行权分配日发生备兑不足情况时交易所锁定量小于备兑量，具体数量根据交易所业务规则计算，与普通现货持仓量、行权分配冻结量有关。若盘中交易所锁定量小于备兑量，投资者需及时买入现货以防备兑仓被上交所强平或深交所转为普通期权持仓
- 行权冻结量ExecVolume为行权日投资者盘中发起看跌期权行权时冻结的股票期权现货持仓量
- 净行权分配量ExecAllocatedVolume依据E日盘后配对结算结果轧差加总的行权交收数量，正值代表收券，负值代表交券，日初和盘中值相同且不变
- 净行权分配金额ExecAllocatedAmount依据E日盘后配对结算结果轧差加总的行权交收金额，正值代表收钱，负值代表交钱，日初和盘中值相同且不变

盘中投资者可以发起的最大行权申请、上交所锁定、深交所备兑数量为股票期权现货可用数量，其计算公式为：

$$股票期权现货可用数量 = 股票期权现货持仓量 - 行权冻结量 - 行权分配冻结量 - 交易所锁定量。$$

5.3.1 股票期权增量模型

增量模型中，股票期权现货持仓量为普通现货持仓中被股票期权业务占用并冻结的数量，这部分持仓只能用于股票期权的行权和备兑业务。当投资者需要更多现货持仓用于股票期权的行权和备兑业务时，需要主动接管普通现货持仓量至股票期权现货持仓量中，同理，当投资者要卖出多余的股票期权现货持仓时，需要主动将股票期权现货持仓归还至普通现货持仓中。各持仓量的关系图如下所示。



根据《中国证券登记结算有限责任公司深圳分公司股票期权试点结算业务指南》中关于行权清算日业务现货持仓冻结的说明，中登要求行权分配冻结量ExecAllocatedFrozenVolume为在E+1日盘中持续冻结行权的认沽期权持仓、被指派的到期备兑持仓以及未到期的备兑持仓对应的现货持仓量之和。

```
1 解锁前一交易日交收锁定的所有备兑证券
2 进行行权有效性检查（即检查行权申报方合约账户中行权合约数量及检查认沽期权行权方日终证券账户标的证券可用数量是否足额），同时对行权的认沽期权合约行权需要的标的证券进行锁定
3 对所有有效行权申报（含认购和认沽）进行指派
4 行权指派后，对未申报行权及未被指派的到期合约予以注销
5 对未被指派行权的合约对应的结算参与人资金保证金账户内维持保证金予以释放
6 对被指派的到期备兑开仓合约对应备兑证券与当日日终所有未到期备兑开仓合约对应备兑证券进行重新锁定
```

按照《中国证券登记结算有限责任公司关于上海证券交易所股票期权试点结算规则》规定：本公司根据检查结果，按照"按比例分配"和"按尾数大小分配"原则，对有效行权与被行权方进行行权指派，将行权指派结果发送结算参与人，并在认沽期权行权方合约账户对应的证券账户中锁定行权交割所需合约标的。**被行权方结算参与人被指派合约对应的维持保证金不予释放。**应付合约标的结算参与人无法交付合约标的的，**本公司对未交付部分按照合约标的当日收盘价的110%进行现金结算。**

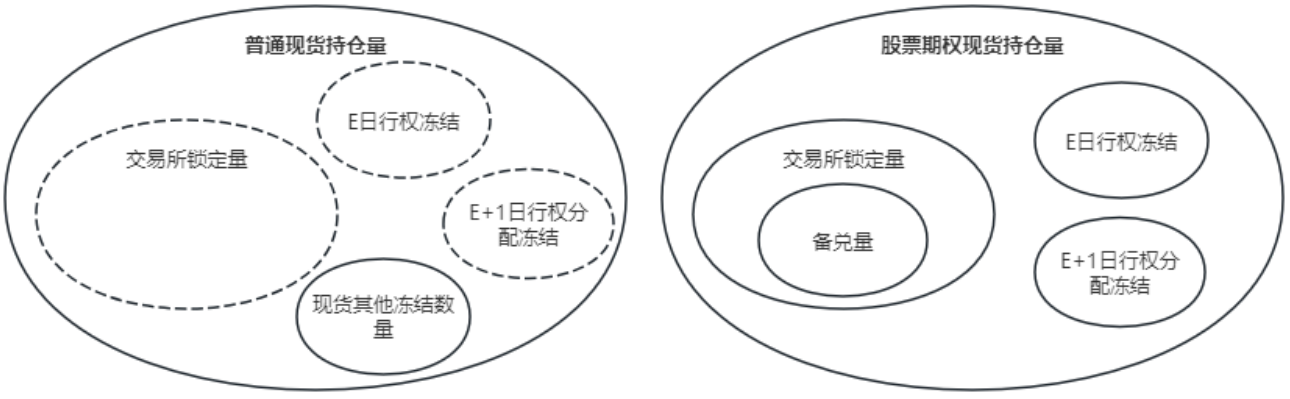
虽然中登行规定了行权分配冻结金额ExecAllocatedFrozenAmount的计算方法，但是有些证券公司在不低于中登监管规定的前提下，根据实际业务开展情况制定行权分配冻结金额的业务规则，易达根据每家证券公司的业务规则计算行权分配冻结金额。由于证券公司对其业务规则的保密要求，具体计算方法不在此罗列，请投资者咨询证券公司获得具体计算方法。

根据本模型的定义，日初股票期权现货持仓量应该等于日初时现货系统中认为初始被股票期权系统接管的持仓量，当普通现货持仓量充足时其值应等于备兑量和行权分配冻结量之和，当普通现货持仓量不足以覆盖备兑持仓和行权分配冻结的现货持仓量时，其值会根据现货系统的规则相应调整，使得其与日初时现货系统中认为初始被股票期权系统接管的持仓量相同。当投资者盘中接管现货持仓时，将减少盘中普通现货持仓量并相应增加盘中股票期权现货持仓量，接管数量不可超过现货系统的可接管数量，其计算公式为$可接管数量 = 普通现货持仓量 - 现货其他冻结数量 - 股票期权现货持仓量$。

5.3.2 股票期权全量模型

全量模型中，股票期权现货持仓量是普通现货持仓量的镜像，通过双向同步机制保持两者一致，定时将普通现货持仓量同步至股票期权现货持仓量，将股票期权系统的交易所锁定量、行权冻结量和行权分配冻结之和同步至现货系统。全量模型存在监管风险和适用面较窄两个问题：

- 由于股票期权系统的交易所锁定量是在业务发生后才同步到现货系统，两者存在时间差，在这时间差内现货持仓已经在交易所被占用但是现货系统认为现货持仓仍未被锁定，若此时投资者在现货系统卖出这部分持仓，现货系统将无法有效拦截该卖出报单而发送至交易所，后被交易所拒绝为错单。这种情况会被监管视为柜台钱仓管理存在漏洞
- 大多数现货柜台都不支持持仓总量读取和锁定绝对数量的功能，这就限制了全量模型无法在大多数场景中使用



行权分配冻结量ExecAllocatedFrozenVolume的公式为：

$行权分配冻结量 = -min(净行权分配量, 0)$。

行权分配冻结量ExecAllocatedFrozenVolume按照期货公司通行的业务规则定义，其计算公式为：

$净行权分配金额 = min(净行权交收金额 + 净差券违约金, 净维持保证金)$

上述公式中各项定义如下：

- 净行权交收金额是依据E日盘后配对结算结果计算的原始交收金额，正值代表收钱，负值代表交钱
- 净差券违约金是为预防交券违约而预先冻结的资金，负值代表需要冻结违约金，0为无违约金，不可能为正值。在E日盘后的配对结算结果中，$差券数量 = max((应付券 - 应收券) - (总券数 - 冻结券数), 0)$，则 $净差券违约金 = -差券数量 \times 现券E日收盘价 \times (1 + 10\%) \times (1 + 10\%)$，其中第一个10%是监管要求，第二个10%在违约情况下无法确定是哪个合约违约，因此只能按照上限进行冻结
- 净维持保证金为期权合约在E+1日盘中按照监管要求冻结的资金，负值代表需要冻结保证金，0为无保证金，不可能为正值

易达提供了现货柜台连接器实现盘中实时双向持仓同步。若经纪商启用了现货柜台连接器，API会通过以下回调函数通知现货持仓同步处于激活状态，该通知消息是持续发送的，若30秒内没有收到通知，则表示同步机制中断，请及时联系经纪商排查问题。

```
virtual void notifySpotAlive(const YDExchange *pExchange)
```

在同步激活状态下，若现货柜台上的持仓发生变化，ydApi和ydExtendedApi会通过以下回调收到通知，newPosition为更新后的现货持仓量：

```
virtual void notifySpotPosition(const YDInstrument *pInstrument,const YDAccount *pAccount,int newPosition)
```

若投资者使用了YDExtendedListener，那么在任何notifySpotPosition回调的时候都会同时回调notifyExtendedSpotPosition：

```
virtual void notifyExtendedSpotPosition(const YDExtendedSpotPosition *pSpotPosition)
```

若没有配置同步机制或者在同步机制盘中失效时，易达股票期权柜台会跳过检查锁仓、备兑开仓、看跌期权行权的现货持仓量，这可能会造成以下问题：

- 锁仓时若实际可锁仓位不足的，交易所会返回锁定错误
- 备兑开仓时若实际可用现货仓位不足的，交易所会返回开仓错误
- 看跌期权行权时实际可用现货仓位不足的，**交易所不会检查现货仓位且会接受行权指令**，这会导致投资者误以为现货仓位足够。但实际上结算时该行权指令会失败，从而使得投资者蒙受损失

5.4 查询持仓

本节内容主要介绍ydExtendedApi中查询实时持仓的方法，所有查询都是基于ydExtendedApi的本地数据执行的，不会到柜台端去执行查询操作，由于所有的查询操作都有加锁操作，性能上会有一定损失。所有查询方法都必须在notifyCaughtUp以后调用，否则会因为报单成交数据不完整而导致持仓数据不准确。要查询日初持仓请参考日初持仓和日初传统组合持仓。

5.4.1 查询衍生品持仓

要获取单个期货与期权持仓可以调用getExtendedPosition方法，返回的YDExtendedPosition结构请参考衍生品持仓模型。

```
virtual const YDExtendedPosition *getExtendedPosition(int positionDate,int positionDirection,int hedgeFlag,const YDInstrument *pInstrument,const YDAccount *pAccount=NULL,bool create=false)
```

调用上述方法的参数说明如下：

参数	说明
positionDate	要查询的持仓的持仓日期 YD_PSD_History：昨仓 YD_PSD_Today：今仓 若交易所不区分今仓和昨仓，请使用YD_PSD_History
positionDirection	要查询的持仓的持仓方向 YD_PD_Long：多头持仓 YD_PD_Short：空头持仓
hedgeFlag	要查询的持仓的投保标志，期货交易所与股票期权交易所的定义不同。 期货交易所如下： YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。为便于编写程序，易达为是否支持套利交易和持仓提供了参数化表示，YDExchange.UseArbitragePosition为true时表示该交易所支持套利交易和持仓，否则为不支持 YD_HF_Hedge=3：套保 股票期权交易所如下： YD_HF_Normal=1：普通 YD_HF_Covered=3：备兑
pInstrument	要查询的合约的指针
pAccount	填写NULL，表示使用当前API登录账户
create	当找不到持仓时是否创建空持仓。 若为true，当找不到持仓时会返回一条新初始化的持仓量为0的持仓；若为false，当找不到持仓时会返回NULL

易达提供了下述两种不同的批量查询持仓的方法，其具有相同的查询参数。

第一种方法需要投资者事先分配固定长度的YDExtendedPosition指针数组，如果预分配的长度不足以容纳的，只会填入预分配数组长度的持仓。无论预分配长度是否足够，本方法的返回值都是返回满足查询条件的持仓的总数，投资者可以利用这个特点，可以调用findExtendedPositions(pFilter, 0, NULL)来快速获取满足条件的持仓的总数。

- 本方法的好处是在持仓不多而且预分配数组长度足够时，可以重复利用预分配的指针数组，而不用每次查询都分配
- 本方法的缺点是如果持仓较多时，可能需要两次调用（第一次获取总持仓数，第二次分配可容纳所有持仓的数组后再次调用）才能完整获取所有满足条件的持仓

第二种方法可以帮助投资者分配能容纳所有满足条件的持仓的空间，但是需要投资者在使用完成后主动调用YDQueryResult.destroy()销毁分配的空间，不可使用delete删除。相比起第一种方法：

- 本方法的好处是调用一次就返回所有的持仓
- 本方法的缺点是需要投资者主动释放由本方法分配的空间，而且每次调用都会分配新的空间，频繁的分配与释放对缓存非常不友好。

```
1 /// positions must have spaces of count, return real number of positions(may be greater than
  count). Only partial will be set if no enough space
2 virtual unsigned findExtendedPositions(const YDExtendedPositionFilter *pFilter,unsigned
  count,const YDExtendedPosition *positions[])
3
4 /// User should call destroy method of return object to free memory after using following
  method
5 virtual YDQueryResult<YDExtendedPosition> *findExtendedPositions(const
  YDExtendedPositionFilter *pFilter)
```

参数YDExtendedPositionFilter各个字段填写说明如下：

字段	说明
PositionDate	要查询的持仓的持仓日期，设置为-1表示该参数不生效 YD_PSD_History：昨仓 YD_PSD_Today：今仓
PositionDirection	要查询的持仓的持仓方向，设置为-1表示该参数不生效 YD_PD_Long：多头持仓 YD_PD_Short：空头持仓
HedgeFlag	要查询的持仓的投保标志，期货交易所与股票期权交易所的定义不同。设置为-1表示该参数不生效 期货交易所如下： YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。为便于编写程序，易达为是否支持套利交易和持仓提供了参数化表示，YDExchange.UseArbitragePosition为true时表示该交易所支持套利交易和持仓，否则为不支持 YD_HF_Hedge=3：套保 股票期权交易所如下： YD_HF_Normal=1：普通 YD_HF_Covered=3：备兑
pInstrument	合约指针，设置为NULL则不限制
pProduct	产品指针，设置为NULL则不限制
pExchange	交易所指针，设置为NULL则不限制
pAccount	投资者请始终设置为NULL

判断某持仓是否符合查询条件的伪代码逻辑如下所示：

```
1  if YDExtendedPositionFilter.PositionDate>=0 and
   YDExtendedPosition.PositionDate!=YDExtendedPositionFilter.PositionDate:
2      return false;
3
4  if YDExtendedPositionFilter.PositionDirection>=0 and
   YDExtendedPosition.PositionDirection!=YDExtendedPositionFilter.PositionDirection:
5      return false;
6
7  if YDExtendedPositionFilter.HedgeFlag>=0 and
   YDExtendedPosition.HedgeFlag!=YDExtendedPositionFilter.HedgeFlag:
8      return false;
9
10 if YDExtendedPositionFilter.pInstrument!=NULL and
   YDExtendedPosition.Instrument!=YDExtendedPositionFilter.pInstrument:
11     return false;
12
13 if YDExtendedPositionFilter.pProduct!=NULL and
   YDExtendedPosition.Product!=YDExtendedPositionFilter.pProduct:
14     return false;
15
16 if YDExtendedPositionFilter.pExchange!=NULL and
   YDExtendedPosition.Exchange!=YDExtendedPositionFilter.pExchange:
17     return false;
18
```

```
19  return true;
```

5.4.2 查询现货持仓

要获取单个股票期权现货持仓可以调用getExtendedSpotPosition方法，返回的YDExtendedHolding结构和含义请参考现货持仓模型。

```
1  virtual const YDExtendedHolding *getExtendedHolding(const YDInstrument *pInstrument,const
   YDAccount *pAccount=NULL,bool create=false)
```

调用上述方法的参数说明如下：

参数	说明
pInstrument	要查询的合约的指针
pAccount	填写NULL，表示使用当前API登录账户
create	当找不到持仓时是否创建空持仓。 若为true，当找不到持仓时会返回一条新初始化的持仓量为0的持仓；若为false，当找不到持仓时会返回NULL

易达提供了下述两种不同的批量查询持仓的方法，其具有相同的查询参数。

第一种方法需要投资者事先分配固定长度的YDExtendedHolding指针数组，如果预分配的长度不足以容纳的，只会填入预分配数组长度的持仓。无论预分配长度是否足够，本方法的返回值都是返回满足查询条件的持仓的总数，投资者可以利用这个特点，可以调用findExtendedHoldings(pFilter, 0, NULL)来快速获取满足条件的持仓的总数。

- 本方法的好处是在持仓不多而且预分配数组长度足够时，可以重复利用预分配的指针数组，而不用每次查询都分配
- 本方法的缺点是如果持仓较多时，可能需要两次调用（第一次获取总持仓数，第二次分配可容纳所有持仓的数组后再次调用）才能完整获取所有满足条件的持仓

第二种方法可以帮助投资者分配能容纳所有满足条件的持仓的空间，但是需要投资者在使用完成后主动调用YDQueryResult.destroy()销毁分配的空间，不可使用delete删除。相比起第一种方法：

- 本方法的好处是调用一次就返回所有的持仓
- 本方法的缺点是需要投资者主动释放由本方法分配的空间，而且每次调用都会分配新的空间，频繁的分配与释放对缓存非常不友好。

```
1  /// holdings must have sapces of count, return real number of holdings(may be greater than
   count). Only partial will be set if no enough space
2  virtual unsigned findExtendedHoldings(const YDExtendedHoldingFilter *pFilter,unsigned
   count,const YDExtendedHolding *holdings[])
3
4  /// User should call destroy method of return object to free memory after using following
   method
5  virtual YDQueryResult<YDExtendedHolding> *findExtendedHoldings(const YDExtendedHoldingFilter
   *pFilter)
```

参数YDExtendedHoldingFilter各个字段填写说明如下：

字段	说明
pInstrument	合约指针。如设置为NULL则不限制。
pProduct	产品指针。如设置为NULL则不限制。

字段	说明
pExchange	交易所指针。如设置为NULL则不限制。
pAccount	投资者请始终设置为NULL

判断某持仓是否符合查询条件的伪代码逻辑如下所示：

```
if YDExtendedHoldingFilter.pInstrument!=NULL and
YDExtendedHolding.Instrument!=YDExtendedHoldingFilter.pInstrument:
    return false;

if YDExtendedHoldingFilter.pProduct!=NULL and
YDExtendedHolding.Product!=YDExtendedHoldingFilter.pProduct:
    return false;

if YDExtendedHoldingFilter.pExchange!=NULL and
YDExtendedHolding.Exchange!=YDExtendedHoldingFilter.pExchange:
    return false;

return true;
```

5.4.3 查询股票期权现货持仓

要获取单个股票期权现货持仓可以调用getExtendedSpotPosition方法，返回的YDExtendedPosition结构和含义请参考股票期权现货持仓模型。

```
virtual const YDExtendedSpotPosition *getExtendedSpotPosition(const YDInstrument
*pInstrument,const YDAccount *pAccount=NULL,bool create=false)
```

调用上述方法的参数说明如下：

参数	说明
pInstrument	要查询的合约的指针
pAccount	填写NULL，表示使用当前API登录账户
create	当找不到持仓时是否创建空持仓。 若为true，当找不到持仓时会返回一条新初始化的持仓量为0的持仓；若为false，当找不到持仓时会返回NULL

易达提供了批量查询持仓的方法，本方法可以帮助投资者分配能容纳所有满足条件的持仓的空间，但是需要投资者在使用完成后主动调用YDQueryResult.destroy()销毁分配的空间。

```
/// User should call destroy method of return object to free memory after using following
method
virtual YDQueryResult<YDExtendedSpotPosition> *findExtendedSpotPositions(const
YDExtendedSpotPositionFilter *pFilter)
```

参数YDExtendedSpotPositionFilter各个字段填写说明如下：

字段	说明
pInstrument	合约指针。如设置为NULL则不限制。

字段	说明
pProduct	产品指针。如设置为NULL则不限制。
pExchange	交易所指针。如设置为NULL则不限制。
pAccount	投资者请始终设置为NULL

判断某持仓是否符合查询条件的伪代码逻辑如下所示：

```
if YDExtendedSpotPositionFilter.pInstrument!=NULL and
YDExtendedSpotPosition.Instrument!=YDExtendedSpotPositionFilter.pInstrument:
    return false;

if YDExtendedSpotPositionFilter.pProduct!=NULL and
YDExtendedSpotPosition.Product!=YDExtendedSpotPositionFilter.pProduct:
    return false;

if YDExtendedSpotPositionFilter.pExchange!=NULL and
YDExtendedSpotPosition.Exchange!=YDExtendedSpotPositionFilter.pExchange:
    return false;

return true;
```

5.4.4 查询传统组合持仓明细

易达为查询上交所和深交所的传统组合持仓明细提供了以下方法，其中combPositionDetailID是唯一确定组合明细的主键：

```
/// getCombPositionDetail can only be used in SSE/SZSE
virtual const YDExtendedCombPositionDetail *getCombPositionDetail(int combPositionDetailID)
```

> **Note**
>
> 易达API没有提供查询传统组合持仓的方法，需要投资者根据日初传统组合持仓自行维护。

返回的YDExtendedCombPositionDetail结构体的字段如下所示：

字段	说明
m_pAccount	传统组合持仓明细属于的投资者的指针
m_pCombPositionDef	传统组合持仓定义指针
Position	传统组合持仓明细量
CombPositionDetailID	传统组合持仓明细ID，只有当在上交所和深交所时有效

易达提供了上述两种不同的批量查询传统组合持仓明细的方法，其具有相同的查询参数：

```
/// combPositionDetails must have spaces of count, return real number of
combPositionDetails(may be greater than count). Only partial will be set if no enough space
virtual unsigned findCombPositionDetails(const YDCombPositionDetailFilter *pFilter,unsigned
count,const YDExtendedCombPositionDetail *combPositionDetails[])

/// User should call destroy method of return object to free memory after using following
method
virtual YDQueryResult<YDExtendedCombPositionDetail> *findCombPositionDetails(const
YDCombPositionDetailFilter *pFilter)
```

判断某传统组合持仓明细是否符合查询条件的伪代码逻辑如下所示，为便于阅读将YDExtendedCombPositionDetail简写为Detail，YDCombPositionDetailFilter简写为Filter：

```
if Filter.IncludeSplit==false and Detail.Position<=0:
    return false

if Filter.pAccount!=NULL and Detail.m_pAccount!=Filter.pAccount:
    return false

if Filter.pCombPositionDef!=NULL and Detail.m_pCombPositionDef!=Filter.pCombPositionDef:
    return false

if Filter.pInstrument!=NULL:
    hasMatchLeg=false
    for (int legIDlegID<2;legID++):
        if matchLeg(Detail->m_pCombPositionDef,legID,Filter):
            hasMatchLeg=true
            break
    if hasMatchLeg==false:
        return false

return true

bool matchLeg(YDCombPositionDef Def,int legID,YDCombPositionDetailFilter Filter):
    if Def.m_pInstrument[legID]!=Filter.pInstrument:
        return false

    if Filter.PositionDirection!=0 and
Filter.PositionDirection!=Def.PositionDirection[legID]:
        return false

    return true
```

参数YDExtendedPositionFilter各个字段填写说明如下：

字段	说明
pCombPositionDef	传统组合持仓定义指针，设置为NULL则不限制
pInstrument	合约指针，设置为NULL则不限制
PositionDirection	要查询的传统组合持仓明细的持仓方向，设置为0表示该参数不生效 YD_PD_Long：多头持仓 YD_PD_Short：空头持仓
IncludeSplit	是否包含已经解组的传统组合持仓明细

第一种方法需要投资者事先分配固定长度的YDExtendedCombPositionDetail指针数组，如果预分配的长度不足容纳的，只会填入预分配数组长度的传统组合持仓明细。无论预分配长度是否足够，本方法的返回值都是返回满足查询条件的传统组合持仓明细的总数，投资者可以利用这个特点，可以调用findCombPositionDetails(pFilter, 0, NULL)来快速获取满足条件的传统组合持仓明细的总数。

- 本方法的好处是在传统组合持仓明细不多而且预分配数组长度足够时，可以重复利用预分配的指针数组，而不用每次查询都分配
- 本方法的缺点是如果传统组合持仓明细较多时，可能需要两次调用（第一次获取总传统组合持仓明细数，第二次分配可容纳所有传统组合持仓明细的数组后再次调用）才能完整获取所有满足条件的传统组合持仓明细

第二种方法可以帮助投资者分配能容纳所有满足条件的传统组合持仓明细的空间，但是需要投资者在使用完成后主动调用YDQueryResult.destroy()销毁分配的空间。相比起第一种方法：

- 本方法的好处是调用一次就返回所有的传统组合持仓明细
- 本方法的缺点是需要投资者主动释放由本方法分配的空间，而且每次调用都会分配新的空间，频繁的分配与释放对缓存非常不友好

6 保证金模型

2023年起，各交易所已经或者即将推出新的组合保证金模型，为了区别，易达将过去的保证金模型称为传统保证金模型。经分析各交易所目前给出的方案，各交易所在很长一段时间内将保持传统保证金和组合保证金共存，即对于同一个投资者，可能存在部分产品使用传统保证金模型，而其他产品使用组合保证金模型。因此，易达建立了保证金模型的概念，并将保证金模型应用到投资者、产品这一层级，即采用相同保证金模型的产品可一起参与对冲、组合等保证金优惠。

目前易达支持以下保证金模型：

- YD_MM_Normal=0：传统保证金模型
- YD_MM_SPBM=1：郑商所SPBM组合保证金模型
- YD_MM_RULE=2：大商所RULE组合保证金模型
- YD_MM_SPMM_SHFE=3：上期所SPMM组合保证金模型
- YD_MM_SPMM_INE=4：能源所SPMM组合保证金模型
- YD_MM_RCAMS=5：中金所RCAMS组合保证金模型

对于使用ydApi的投资者，易达并没有提供判断某个产品或合约属于的交易模型，需要投资者自行从组合保证金参数中解析。对于使用ydExtendedApi的投资者，可以使用以下方法获得某个合约或产品属于的保证金模型：

```
virtual int getMarginModel(const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
virtual int getMarginModel(const YDProduct *pProduct,const YDAccount *pAccount=NULL)
```

6.1 传统保证金模型

易达的期货交易所保证金收取和优惠原则是尽可能**与主流主席柜台系统保持一致**。需要特别说明的是，易达和主流主席柜台系统一样对上期所、能源所和中金所的挂单与持仓一起合并计算大边保证金，但是目前在以下两点和主流主席柜台系统在业务规则上不一致：

- 大商所期权组合保证金，易达与主流主席柜台系统的计算方法不一样
- 郑商所对锁单，易达把持仓和挂单一起按照大边计算优惠了保证金，主流主席柜台系统目前只计算了持仓大边优惠而没有计算挂单的大边优惠

因此，除了上述两个业务例外之外，在交易所休市阶段易达和主流主席柜台系统 的保证金授权应该是完全一致的，在有交易的时候，会因为交易所回报顺序、行情更新时机、柜台重算时机不同，而导致易达柜台和主流主席柜台系统有细微的差异。

易达并不区分报单冻结保证金和持仓保证金，而是统一当做保证金处理，计算大边保证金时也是把报单保证金和持仓保证金统一加总后计算大边的。为方便投资者理解，本文区分了报单保证金和持仓保证金，但是在柜台内部是不区分的。

6.1.1 期货保证金

期货多头保证金计算公式为：

$$\text{期货多头持仓保证金} = (\text{价格} \times \text{合约乘数} \times \text{基于金额的多头保证金率} + \text{基于手数的多头保证金率}) \times \text{数量}$$

期货空头保证金计算公式为：

$$\text{期货空头持仓保证金} = (\text{价格} \times \text{合约乘数} \times \text{基于金额的空头保证金率} + \text{基于手数的空头保证金率}) \times \text{数量}$$

对于**报单**，计算期货报单保证金时使用的数量为报单量，价格受到YDSystemParam中（Name, Target）为（OrderMarginBasePrice, Futures）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价。
- YD_CBT_OrderPrice：当报单为市价单时，价格为涨停板价，当报单为限价单/FAK/FOK时，价格为报单价格。这是默认配置的方式，也是主流主席柜台系统当前使用的方式

- YD_CBT_SamePrice：与期货持仓保证金率采用相同的价格，即使用YDSystemParam中（Name, Target）为（MarginBasePrice, Futures）的参数值。若此时期货持仓保证金率使用的是YD_CBT_OpenPrice时，期货报单保证金使用的价格是昨结算价

对于**昨仓**，计算期货持仓保证金时使用的数量为持仓量，价格始终使用昨结算价。此处的昨仓是指按照逐日盯市规则结算后剩余的持仓，并非衍生品持仓模型中今昨仓的概念。

对于**今仓**，计算期货持仓保证金时使用的数量为持仓量，价格受到YDSystemParam中（Name, Target）为（MarginBasePrice, Futures）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价
- YD_CBT_OpenPrice：开仓价，按照开仓价逐个计算持仓明细对应的保证金。这是默认配置的方式，也是主流主席柜台系统当前使用的方式
- YD_CBT_LastPrice：最新价
- YD_CBT_MarketAveragePrice：市场平均成交价
- YD_CBT_MaxLastPreSettlementPrice：最新价和昨结算价的大者。当该合约当日全市场成交量为0时，使用昨结算价。

6.1.2 期权保证金

6.1.2.1 商品期权保证金

适用于上期所、能源所、大商所、广期所、郑商所交易的商品期权。

看涨商品期权的保证金公式如下：

$$\begin{aligned}\text{商品期权看涨保证金} = &[\text{期权价格} \times \text{期权合约乘数} + max(\text{看涨期权基础保证金}-\text{看涨期权虚值}/2,\\ &\text{看涨期权基础保证金}/2)] \times \text{数量}\end{aligned}$$

$$\begin{aligned}\text{看涨期权基础保证金} = &\text{基础合约价格} \times \text{基础合约乘数} \times \text{基于金额的看涨期权空头保证金率}\\ &+\text{基于手数的看涨期权空头保证金率}\end{aligned}$$

$$\text{看涨期权虚值} = max((\text{行权价}-\text{基础合约价格}) \times \text{期权合约乘数}, 0)$$

看跌商品期权的保证金公式如下：

$$\begin{aligned}\text{商品期权看跌保证金} = &[\text{期权价格} \times \text{期权合约乘数} + max(\text{看跌期权基础保证金}-\text{看跌期权虚值}/2,\\ &\text{看跌期权基础保证金}/2)] \times \text{数量}\end{aligned}$$

$$\begin{aligned}\text{看跌期权基础保证金} = &\text{基础合约价格} \times \text{基础合约乘数} \times \text{基于金额的看跌期权空头保证金率}\\ &+\text{基于手数的看跌期权空头保证金率}\end{aligned}$$

$$\text{看跌期权虚值} = max((\text{基础合约价格}-\text{行权价}) \times \text{期权合约乘数}, 0)$$

对于**报单**，计算期权报单保证金时使用的数量为报单量，期权价格受到YDSystemParam中（Name, Target）为（OrderMarginBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式
- YD_CBT_OrderPrice：当报单为市价单时，价格为涨停板价，当报单为限价单/FAK/FOK时，价格为报单价格。
- YD_CBT_SamePrice：与期权持仓保证金率采用相同的价格，即使用YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值。若此时期权持仓保证金率使用的是YD_CBT_OpenPrice时，期权报单保证金使用的价格是昨结算价

对于**昨仓**和**今仓**，计算期权持仓保证金时使用的数量为持仓量，期权价格受到YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价
- YD_CBT_OpenPrice：开仓价，按照开仓价逐个计算持仓明细对应的保证金
- YD_CBT_LastPrice：最新价
- YD_CBT_MarketAveragePrice：市场平均成交价

- YD_CBT_MaxLastPreSettlementPrice：最新价和昨结算价的大者。当该合约当日全市场成交量为0时，使用昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式。

对于**报单**、**昨仓**和**今仓**，基础合约价格受到YDSystemParam中（Name, Target）为（MarginBasePriceAsUnderlying,Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式
- YD_CBT_LastPrice：最新价
- YD_CBT_MaxLastPreSettlementPrice：最新价和昨结算价的大者。当该合约当日全市场成交量为0时，使用昨结算价。

6.1.2.2 股指期权保证金

适用于中金所交易的股指期权。

股指期权看涨保证金的计算公式为：

$$\text{股指期权看涨保证金} = (\text{期权价格} \times \text{期权合约乘数} + max(\text{看涨期权基础保证金} - \text{看涨期权虚值}, \text{最低保障系数} \times \text{看涨期权基础保证金})) \times \text{数量}$$

$$\text{看涨期权基础保证金} = \text{基础合约价格} \times \text{基础合约乘数} \times \text{基于金额的看涨期权空头保证金率}$$

$$\text{看涨期权虚值} = max((\text{行权价} - \text{基础合约价格}) \times \text{期权合约乘数}, 0)$$

股指期权看跌保证金的计算公式为：

$$\text{股指期权看跌保证金} = (\text{期权价格} \times \text{期权合约乘数} + max(\text{看跌期权基础保证金}1 - \text{看跌期权虚值}, \text{最低保障系数} \times \text{看跌期权基础保证金}2)) \times \text{数量}$$

$$\text{看跌期权基础保证金}1 = \text{基础合约价格} \times \text{基础合约乘数} \times \text{基于金额的看跌期权空头保证金率}$$

$$\text{看跌期权基础保证金}2 = \text{行权价} \times \text{基础合约乘数} \times \text{基于金额的看跌期权空头保证金率}$$

$$\text{看跌期权虚值} = max((\text{基础合约价格} - \text{行权价}) \times \text{期权合约乘数}, 0)$$

最低保障系数取自于YDSystemParam中（Name, Target）为（MarginLowerBoundaryCoef,MarginCalcMethod1）的参数值，目前股指期权的最低保障系数应该为0.5。

对于**报单**，计算期权报单保证金时使用的数量为报单量，期权价格受到YDSystemParam中（Name, Target）为（OrderMarginBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式
- YD_CBT_OrderPrice：当报单为市价单时，价格为涨停板价，当报单为限价单/FAK/FOK时，价格为报单价格。
- YD_CBT_SamePrice：与期权持仓保证金率采用相同的价格，即使用YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值。若此时期权持仓保证金率使用的是YD_CBT_OpenPrice时，期权报单保证金使用的价格是昨结算价

对于**昨仓**和**今仓**，计算期权持仓保证金时使用的数量为持仓量，期权价格受到YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价
- YD_CBT_OpenPrice：开仓价，按照开仓价逐个计算持仓明细对应的保证金
- YD_CBT_LastPrice：最新价
- YD_CBT_MarketAveragePrice：市场平均成交价
- YD_CBT_MaxLastPreSettlementPrice：最新价和昨结算价的大者。当该合约当日全市场成交量为0时，使用昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式

对于**报单**、**昨仓**和**今仓**，由于易达柜台盘中不获取股指行情，所以无论如何设置，股指期权的基础合约价格总是用昨结算价。

6.1.2.3 股票期权保证金

在生产实践中，股票期权的保证金模型可以分为线性和非线性两种，易达柜台通过控制计算公式中的参数同时实现了这两个保证金模型。例如，当线性系数为1.2，其他参数为交易所标准值时（目前基础合约保证金率为12%，最低保障系数为7%），即采用线性模式；当基础合约保证金率为15%，最低保障系数为8%，而线性系数为1时，即采用非线性模式。

股票期权看涨保证金的计算公式为：

$$\begin{aligned}\text{股票期权看涨保证金} = \text{线性系数} \times (\text{期权价格} + max(\text{基础合约保证金率} \times \text{基础合约价格} \\ -\text{看涨虚值度}, \text{最低保障系数} \times \text{基础合约价格})) \times \text{期权合约乘数} \times \text{数量}\end{aligned}$$

$$\text{看涨虚值度} = max(\text{行权价} - \text{基础合约价格}, 0)$$

股票期权看跌保证金的计算公式为：

$$\begin{aligned}\text{看跌期权空方的每手保证金} = \text{线性系数} \times (\text{期权价格} + min(max(\text{基础合约保证金率} \times \text{基础合约价格} \\ -\text{看跌虚值度}, \text{最低保障系数} \times \text{行权价}), \text{行权价}-\text{期权最新价})) \times \text{期权合约乘数} \times \text{数量}\end{aligned}$$

$$\text{看跌虚值度} = max(\text{基础合约价格} - \text{行权价}, 0)$$

对于**报单**，计算期权报单保证金时使用的数量为报单量，期权价格受到YDSystemParam中（Name, Target）为（OrderMarginBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式
- YD_CBT_OrderPrice：当报单为市价单时，价格为涨停板价，当报单为限价单/FAK/FOK时，价格为报单价格。
- YD_CBT_SamePrice：与期权持仓保证金率采用相同的价格，即使用YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值。若此时期权持仓保证金率使用的是YD_CBT_OpenPrice时，期权报单保证金使用的价格是昨结算价

对于**昨仓**和**今仓**，计算期权持仓保证金时使用的数量为持仓量，期权价格受到YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价
- YD_CBT_OpenPrice：开仓价，按照开仓价逐个计算持仓明细对应的保证金
- YD_CBT_LastPrice：最新价
- YD_CBT_MarketAveragePrice：市场平均成交价
- YD_CBT_MaxLastPreSettlementPrice：最新价和昨结算价的大者。当该合约当日全市场成交量为0时，使用昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式

对于**报单**、**昨仓**和**今仓**，基础合约价格受到YDSystemParam中（Name, Target）为（MarginBasePriceAsUnderlying,Options）的参数值控制，以下列举不同参数值时系统的处理方式：

- YD_CBT_PreSettlementPrice：昨结算价。这是默认配置的方式，也是主流主席柜台系统当前使用的方式
- YD_CBT_LastPrice：最新价
- YD_CBT_MaxLastPreSettlementPrice：最新价和昨结算价的大者。当该合约当日全市场成交量为0时，使用昨结算价。

6.1.3 期权行权保证金

6.1.3.1 商品期权行权保证金

商品期权的看涨期权保证金的计算公式为：

$$\begin{aligned}\text{商品期权看涨行权保证金} = ((\text{基础合约昨结算价} \times \text{基础合约乘数} \times \text{基础合约的基于金额的多头保证金率} \\ +\text{基础合约的基于手数的多头保证金率}) \times \text{期权基础乘数} \\ +max(\text{行权价} - \text{基础合约昨结算价}, 0) \times \text{期权合约乘数}) \times \text{数量}\end{aligned}$$

商品期权的看跌期权保证金的计算公式为：

$$\begin{aligned}\text{商品期权看跌行权保证金} = &\big((\text{基础合约昨结算价} \times \text{基础合约乘数} \times \text{基础合约的基于金额的空头保证金率} \\ &+\text{基础合约的基于手数的空头保证金率}) \times \text{期权基础乘数} \\ &+max(\text{基础合约昨结算价} - \text{行权价}, 0) \times \text{期权合约乘数}\big) \times \text{数量}\end{aligned}$$

基础乘数为期权合约的UnderlyingMultiply。

6.1.3.2 股票期权行权保证金

股票期权看涨行权保证金的计算公式为：

$$\text{股票期权看涨行权保证金} = \text{行权价} \times \text{期权乘数}$$

股票期权看跌行权不收取保证金。

6.1.4 保证金优惠

交易所优惠保证金的主要手段分为单向大边保证金和组合保证金两种，虽然最终的效果类似，但是其使用方式和复杂程度截然不同，请投资者区别看待两种优惠方式，不可混为一谈。目前中金所、上期所、能源所、郑商所支持大边保证金业务，大商所、郑商所、上交所和深交所支持组合保证金业务。

6.1.4.1 单向大边保证金

郑商所支持**同合约单向大边保证金**，具体方法为分别将属于同一合约的多头和空头的投机、套保仓位和报单的保证金加总后，取保证金较大的一边为该合约上实际收取的保证金。

上期所和能源所支持**同产品单向大边保证金**，具体方法为分别将属于同一产品的多头和空头的投机、套保仓位和报单的保证金加总后，取保证金较大的一边为该产品上实际收取的保证金。

中金所支持**跨产品单向大边保证金**，具体方法为分别将属于同一产品组的多头和空头的投机、套保仓位和报单的保证金加总后，取保证金较大的一边为该产品组上实际收取的保证金。产品组的定义在YDProduct.m_pMarginProduct中，该字段指向了产品组的组长产品（组长产品并不是固定的，可能会随着初始化数据的变化而有所不同），组长产品和所有m_pMarginProduct指向该组长产品的属于同一产品组。目前中金所有两个产品组，所有的股指期货IF、IC、IH、IM属于同一产品组，所有的国债期货TF、TS、T属于同一产品组。

为方便策略程序识别需要参与单向大边保证金计算的合约，YDInstrument.SingleSideMargin为True时表示该合约需要参与单向大边保证金计算，否则不参与。通常情况下上期所、能源所和中金所的期货合约的SingleSideMargin为True，但由于临近交割的期货合约不应参与单向大边保证金计算，单向大边保证金计算中需要剔除这些合约，所以这部分临近交割的合约的SingleSideMargin会被设置为False。此部分设置来自于主流主席柜台系统的日初数据。

6.1.4.2 传统组合保证金

易达为各交易所不同的组合类型制定了不同的组合保证金优惠方法，下面将逐一介绍。

6.1.4.2.1 期货组合

以下组合类型适用本优惠算法：

- YD_CPT_DCE_FuturesOffset：大商所期货对锁
- YD_CPT_DCE_FuturesCalendarSpread：大商所期货跨期
- YD_CPT_DCE_FuturesProductSpread：大商所期货跨品种
- YD_CPT_GFEX_FuturesOffset：广期所期货对锁
- YD_CPT_GFEX_FuturesCalendarSpread：广期所期货跨期
- YD_CPT_GFEX_FuturesProductSpread：广期所期货跨品种
- YD_CPT_CZCE_Spread：郑商所套利

期货组合保证金需要先按照规则选择小边，然后再使用选中小边持仓的保证金作为节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。小边的选择规则为：

- 比较两腿的交易所期货保证金，较小的一边被选中，若相同则继续。交易所期货保证金与期货保证金的计算公式相同，在计算时价格使用昨结算价，保证金率使用交易所保证金率，交易所保证金率可以在YDInstrument.m_pExchangeMarginRate中找到
- 若组合类型为YD_CPT_CZCE_Spread，则直接选择右腿，否则继续
- 比较两腿的交割月，较远月的一边被选中，若相同则继续
- 比较两腿的合约代码，较大的合约代码被选中。合约代码比较是字符串比较

6.1.4.2.2 期权跨式

以下组合类型适用本优惠算法：

- YD_CPT_DCE_OptionsStraddle：大商所卖出期权跨式
- YD_CPT_DCE_OptionsStrangle：大商所卖出期权宽跨式
- YD_CPT_GFEX_OptionsStraddle：广期所卖出期权跨式
- YD_CPT_GFEX_OptionsStrangle：广期所卖出期权宽跨式
- YD_CPT_CZCE_StraddleStrangle：郑商所卖出跨式或宽跨式

上述期权组合保证金需要先按照规则选择小边，然后再使用选中小边计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：$交易所期权保证金 \times 持仓量$，交易所期权保证金与期权保证金的计算公式相同，在计算时价格使用昨结算价，保证金率使用交易所保证金率，交易所保证金率可以在YDInstrument.m_pExchangeMarginRate中找到。小边的选择规则为：

- 比较两腿的期权保证金，较小的一边被选中，若相同则继续
- 计算两腿的节约保证金，选择较小的节约保证金为最终的节约保证金

以下组合类型适用本优惠算法：

- YD_CPT_StockOption_KS：上交所、深交所所跨式空头
- YD_CPT_StockOption_KKS：上交所、深交所宽跨式空头

上述期权组合保证金需要先按照规则选择小边，然后再使用选中小边计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：
$(期权保证金 - 期权昨结算价 \times 期权合约乘数 \times 线性系数) \times 持仓量$。小边的选择规则为：

- 比较两腿的期权保证金，较小的一边被选中，若相同则继续
- 计算两腿的节约保证金，选择较小的节约保证金为最终的节约保证金

6.1.4.2.3 卖权覆盖

以下组合类型适用本优惠算法：

- YD_CPT_DCE_SellOptionsCovered：大商所卖出期权期货组合
- YD_CPT_GFEX_SellOptionsCovered：广期所卖出期权期货组合
- YD_CPT_CZCE_SellOptionConvered：郑商所卖出期权期货组合

直接使用左腿计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：$交易所期权保证金 \times 持仓量$。公式中的期权价格使用YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值指定的价格；交易所期权保证金与商品期权保证金的计算公式相同，在计算时价格使用昨结算价，保证金率使用交易所保证金率，交易所保证金率可以在YDInstrument.m_pExchangeMarginRate中找到。

6.1.4.2.4 买权组合

以下组合类型适用本优惠算法：

- YD_CPT_DCE_OptionsOffset：大商所期权对锁
- YD_CPT_DCE_BuyOptionsVerticalSpread：大商所买入期权垂直套利
- YD_CPT_DCE_BuyOptionsCovered：大商所买入期权期货组合

- YD_CPT_GFEX_OptionsOffset：广期所期权对锁
- YD_CPT_GFEX_BuyOptionsVerticalSpread：广期所买入期权垂直套利
- YD_CPT_GFEX_BuyOptionsCovered：广期所买入期权期货组合

当投资者平上述组合的买腿时是需要加收保证金的，在主流主席柜台系统上若可用资金不足以加收保证金时会禁止平仓。易达认为禁止平仓会影响风险的释放，可能反而会造成更大的风险，同时也违反了易达平仓不检查资金的原则，因此在易达中平买腿时可能会导致可用资金变为负数。易达将选择权力交给经纪商，若经纪商对此处理有异议，可以关闭本组合保证金优惠功能，功能关闭后仍然可以进行组合但是不再优惠保证金，这样就确保平仓不会导致穿仓，若经纪商认可易达的处理方式，可以打开该功能。在初始安装中，该功能是默认关闭的。投资者可以通过查看YDSystemParam中（Name, Target）为（PortfolioMarginConcession, DCELongOptionPortfolio）的参数值了解该功能是否开启。

若开启了优惠，则直接使用右腿计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：$(1 - \text{组合保证金折扣}) \times \text{期权保证金} \times \text{持仓量}$。公式中的组合保证金折扣为YDCombPositionDef.Parameter，详情参见传统组合持仓定义。

6.1.4.2.5 卖权垂直套利

以下组合类型适用本优惠算法：

- YD_CPT_DCE_SellOptionsVerticalSpread：大商所卖出期权垂直套利
- YD_CPT_GFEX_SellOptionsVerticalSpread：广期所卖出期权垂直套利

当投资者平上述组合的买腿时是需要加收保证金的，在主流主席柜台系统上若可用资金不足以加收保证金时会禁止平仓。易达认为禁止平仓会影响风险的释放，进而造成更大的风险，同时也违反了平仓不检查资金的通行做法，因此在易达中平买腿时可能会导致可用资金变为负数。易达将选择权交给经纪商：若经纪商对此处理有异议，可以关闭本组合保证金优惠功能，功能关闭后仍然可以进行组合但是不再优惠保证金，这样就确保平仓不会导致穿仓；若经纪商认可易达的处理方式，可以打开该功能。在初始安装中，该功能是默认关闭的。投资者可以通过查看YDSystemParam中（Name, Target）为（PortfolioMarginConcession, DCELongOptionPortfolio）的参数值了解该功能是否开启。

若开启了优惠，则直接使用左腿计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：

$$max(\text{左腿期权保证金} - |\text{左腿行权价} - \text{右腿行权价}| \times \text{右腿合约乘数}, 0) \times \text{持仓量}$$

6.1.4.2.6 牛熊市价差

以下组合类型适用下面优惠算法：

- YD_CPT_StockOption_CNSJC：上交所、深交所认购牛市价差
- YD_CPT_StockOption_PXSJC：上交所、深交所认沽熊市价差

直接使用右腿计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：$\text{右腿每手保证金} \times \text{持仓量}$。右腿每手保证金即为右腿实际的保证金，具体计算方法请参考股票期权保证金。

以下组合类型适用下面优惠算法：

- YD_CPT_StockOption_CXSJC：上交所、深交所认购熊市价差
- YD_CPT_StockOption_PNSJC：上交所、深交所认沽牛市价差

直接使用右腿计算节约保证金，并从原有两腿的保证金之和中减去节约保证金，即得到组合保证金。节约保证金的计算公式为：

$$(\text{右腿每手保证金} - |\text{左腿行权价} - \text{右腿行权价}| \times \text{右腿合约乘数} \times \text{右腿线性系数}) \times \text{传统组合持仓量}$$

右腿每手保证金即为右腿实际的保证金，具体计算方法请参考股票期权保证金。

6.1.5 保证金试算

保证金以及组合保证金优惠的计算比较复杂，为了方便使用ydExtendedApi的投资者以指定的价格预算保证金，易达提供了一些列方法供使用。

6.1.5.1 报单保证金试算

投资者可以随时通过下列方法获取每手报单的期货保证金，试算过程没有计算大边保证金，仅仅是该报单自身的保证金。本方法不适用组合合约。

```
virtual double getMarginPerLot(const YDInstrument *pInstrument,int hedgeFlag,int anyDirection,double openPrice,const YDAccount *pAccount=NULL)
```

上述方法中的参数如下表：

参数	说明
pInstrument	合约指针
hedgeFlag	投保标志。 YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。 YD_HF_Hedge=3：套保
anyDirection	买卖方向，可以使用YD_D_Buy或YD_PD_Long表示买，YD_D_Sell或YD_PD_Short表示卖
openPrice	根据YDSystemParam中（Name, Target）为（MarginBasePrice, Options）的参数值决定使用的价格： YD_CBT_PreSettlementPrice：始终使用昨结算价，openPrice无效 YD_CBT_OpenPrice：使用openPrice指定的值 YD_CBT_LastPrice：始终使用最新价，openPrice无效 YD_CBT_MarketAveragePrice：始终使用市场平均成交价，openPrice无效 YD_CBT_MaxLastPreSettlementPrice：通常使用最新价和昨结算价的大者，当该合约当日全市场成交量为0时，使用昨结算价。openPrice无效
pAccount	设置为NULL

投资者可以随时通过下列方法获取每手报单的期权保证金，试算使用的价格是由API根据价格设置自行选择的。

```
virtual double getOptionsShortMarginPerLot(const YDInstrument *pInstrument,int hedgeFlag,bool includePremium,const YDAccount *pAccount=NULL)
```

上述方法中的参数如下表：

参数	说明
pInstrument	合约指针
hedgeFlag	投保标志，期货交易所与股票期权交易所的定义不同。 期货交易所如下： YD_HF_Speculation=1：投机 YD_HF_Arbitrage=2：套利，只有中金所支持。 YD_HF_Hedge=3：套保 股票期权交易所如下： YD_HF_Normal=1：普通 YD_HF_Covered=3：备兑

参数	说明
includePremium	是否需要包含权利金
pAccount	设置为NULL

6.1.5.2 持仓保证金试算

以下方法可以试算在不同的价格条件下持仓的每手保证金，如果需要获取持仓完整的保证金需要自行乘以持仓量，对于期货持仓不计算大边保证金。与报单保证金试算不同，此处的openPrice并不受到YDSystemParam的影响，在计算时标的价格直接使用的是参数中openPrice指定的价格。

```
virtual double getMarginPerLot(const YDExtendedPosition *pPosition,double openPrice)
```

6.1.5.3 组合保证金试算

以下方法可以试算尚不存在的组合，试算使用的价格是昨结算价，所以可能会与组合后实际收取的保证金有所区别。

```
virtual double getCombPositionMarginPerLot(const YDCombPositionDef *pCombPositionDef,const
YDAccount *pAccount=NULL)
```

以下方法可以获取持仓明细真实的两腿保证金、节约保证金和组合保证金，
$组合保证金 = 左腿保证金 + 右腿保证金 - 节约保证金$，其中左右腿保证金分别是legMargins[0]和legMargins[1]，节约保证金是本方法的返回值。

```
virtual double getCombPositionMarginSaved(const YDExtendedCombPositionDetail
*pCombPositionDetail,double legMargins[])
```

6.2 组合保证金模型

由于各交易所已公布保证金算法的具体逻辑，本文不再讨论，详情请参见交易所相关文档。

易达组合保证金模型与CTP保持一致，以下将介绍组合保证金模型中的通用概念以及与交易所组合保证金模型的差异。

6.2.1 组合保证金模型通用概念

6.2.1.1 投资者保证金系数

组合保证金模型中，投资者保证金系数可以设置在组保模型上，可在YDAccountMarginModelInfo.MarginRatio获取组保模型系数，该系统盘中可修改，详情参见账户组合保证金参数。

从`1.486`版本开始也可以设置在组保模型的产品群上，可在YDAccountProductGroupMarginModelParam.MarginRatio获取组保模型的产品群系数，该系数不可盘中修改，详情参见产品群组合保证金参数。

投资者在某组合保证金模型的保证金公式为：

$$投资者保证金 = 交易所保证金 \times 投资者组保模型系数$$

$$交易所保证金 = \sum_{产品群} 产品群交易所保证金 \times 投资者产品群系数$$

6.2.1.2 平仓校验

部分组合保证金模型中，平仓时可能需要加收保证金，若此时可用资金不足以覆盖平仓加收保证金，就会产生资金透支，为此易达提供了平仓校验模式。

平仓校验模式用于控制易达柜台收到平仓报单时是否检查可用资金，该参数从CTP日初数据中转换而来，可在YDAccountMarginModelInfo.CloseVerify获取，盘中可实时上场，详情参见账户组合保证金参数。目前支持的模式如下：

- YD_CV_NotVerify：不检查

- YD_CV_Verify：检查。检查逻辑如下：
  - 如果造成可用资金增加，无论此时可用资金是否小于0，则通过，否则继续判断
  - 如果组合保证金模型针对当前情况特别要求不检查，则通过，否则继续判断
  - 如果扣除加收保证金后可用资金大于等于0，则通过，否则不通过

6.2.1.3 适用范围

部分组合保证金模型中，可设置该组合保证金模型的产品子集作为投资者的适用范围，例如，适用SPBM的产品包括MA、PF、SR，但是可以设置某投资者仅使用MA和PF两个产品。适用范围从CTP日初数据中转换而来，可在YDAccountMarginModelInfo.ProductRange获取，盘中不可实时上场，详情参见账户组合保证金参数。

属于组合保证金模型的产品，但是该投资者不适用的产品，仍然使用传统保证金模型计算该产品的保证金。

6.2.2 郑商所SPBM

在SPBM标准模型的基础上，根据CTP主席柜台的业务逻辑相应修订如下。

6.2.2.1 冻结附加保证金

按照SPBM标准模型，报单时可能减少保证金。为避免此类问题，当某产品族的品种内合约间对锁仓费率折扣比例IntraRateY<1时，加收冻结附加保证金。冻结附加保证金的计算公式为：

$$\text{冻结附加保证金} = (\text{品种内对冲保证金} - min(\text{买实仓保证金，卖实仓保证金})) \times (1 - IntraRateY)$$

上述买实仓保证金和卖实仓保证金为SPBM标准模型中，计算品种买保证金、卖保证金时仅包含持仓的部分。

与上述情况类似，当某产品族的品种内对锁仓费率折扣比例IntraRateZ<0.5时也会产生类似问题，但考虑到计算比较复杂，且目前SPBM生产参数中没有这种情况，因此暂不实现。

6.2.2.2 平仓校验

在平仓校验时，本组合保证金模型对于期权多头平仓不要求校验。

6.2.2.3 行权保证金

在行权申请和放弃自动行权时，需将对应的持仓从买方期权冲抵价值中扣除，扣除方式等同于该期权已平仓。对于行权申请，需同时冻结行权保证金。

$$\text{行权保证金} = \text{行权手数} \times (\text{基础期货每手保证金} + \text{期权虚值})$$

$$\text{基础期货每手保证金} = \text{投资者保证金系数} \times \text{合约乘数} \times \text{昨结算价} \times (\text{品种保证金标准} + \text{合约提高保证金标准})$$

期权虚值的计算公式请参考期权保证金章节的内容，SPBM的行权保证金计算中期权虚值始终使用昨结算价。

6.2.3 大商所Rule

在Rule标准模型的基础上，根据CTP主席柜台的业务逻辑相应修订如下。

6.2.3.1 行权保证金

在行权申请和放弃自动行权时，按照平仓冻结方式重新计算组合保证金。每手行权保证金为0。

6.2.4 上期所和能源所SPMM

6.2.4.1 平仓冻结保证金

平仓冻结保证金按照每个商品组分别计算，首先按照多空头分别计算商品组内的平仓报单纯粹价格风险，计算方式同成交，然后累加获得商品组多头平仓报单纯粹价格风险之和（卖平报单）和商品组空头平仓报单纯粹价格风险之和（买平报单），最后根据下面公式获得单商品组的平仓冻结保证金，将所有商品组的平仓冻结保证金累加后获得投资者总平仓冻结保证金，总平仓冻结保证金作为SPMM投资者保证金的加项计入最终保证金。

商品组平仓冻结保证金 $= max(InCommPref - 1, 0) * [min(L1, S1) - min(L1 - L2, S1 - S2)]$

上述公式中的符号定义如下：

- InCommPref：跨期优惠系数
- L1：商品组多头持仓纯粹价格风险
- S1：商品组空头持仓纯粹价格风险
- L2：商品组多头平仓报单纯粹价格风险之和（卖平报单）
- S2：商品组空头平仓报单纯粹价格风险之和（买平报单）

6.2.5 中金所RCAMS

不支持套利户的组合持仓，如果投资者申请了套利户的组合持仓，会被柜台自动丢弃。

7 交易

7.1 报单

报单指令是最常用的指令，投资者可以调用报单指令发起期货和期权的交易。可以用于报单的指令如下所示。

```
virtual bool insertOrder(YDInputOrder *pInputOrder,const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
virtual bool insertMultiOrders(unsigned count,YDInputOrder inputOrders[],const YDInstrument *instruments[],const YDAccount *pAccount=NULL)

// only available in ydExtendedApi
virtual bool checkAndInsertOrder(YDInputOrder *pInputOrder,const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
virtual bool checkOrder(YDInputOrder *pInputOrder,const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
```

7.1.1 普通报单

首先介绍最普通的报单指令insertOrder，如果该函数返回false，则说明到柜台的网络发生了中断。

```
virtual bool insertOrder(YDInputOrder *pInputOrder,const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
```

衍生品报单的参数说明如下：

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Normal
	OrderRef	客户报单编号，柜台在回报中会带回客户报单编号，投资者可以以此将报单和回报对应起来。
	Direction	YD_D_Buy：买 YD_D_Sell：卖
	OffsetFlag	对于**普通合约**： YD_OF_Open：开仓 YD_OF_Close：平仓，适用于非上期所和能源所。若在上期所和能源所使用会被视作YD_OF_CloseYesterday YD_OF_CloseYesterday：平昨仓，适用于上期所和能源所。若在非上期所和能源所使用会被视作YD_OF_Close YD_OF_CloseToday：平今仓，适用于上期所和能源所。若在非上期所和能源所使用会被视作YD_OF_Close 对于大商所、广期所、郑商所的**组合合约**： YD_OF_Open：同时开左右腿 YD_OF_Close：同时平左右腿 YD_OF_Open1Close2：开左腿平右腿 YD_OF_Close1Open2：平左腿开右腿
	HedgeFlag	YD_HF_Speculation：投机 YD_HF_Arbitrage：套利，只支持中金所 YD_HF_Hedge：套保

参数	字段	说明
	OrderType	YD_ODT_Limit：限价单 YD_ODT_Market：市价单 YD_ODT_FAK：FAK单 YD_ODT_FOK：FOK单
	Price	填写价格，价格不能超过涨跌停板限制，易达不控制价格波动带
	OrderVolume	开平仓数量，受到以下限制： 不能超过合约的MaxMarketOrderVolume、MaxLimitOrderVolume 不能低于合约的MinMarketOrderVolume、MinLimitOrderVolume 不能超过持仓量限制，注意上交所和深交所需去除组合仓以后的持仓量
	OrderTriggerType	触发类型 YD_OTT_NoTrigger：无触发条件 YD_OTT_TakeProfit：止盈触发 YD_OTT_StopLoss：止损触发 目前支持大商所和广期所的触发单
	TriggerPrice	触发价格
	ExchangeOrderAttribute	交易所报单属性，由报单所属交易所解释其含义。 YD_EOA_DCE_GIS=1：大商所GIS当前小节有效报单标志
YDInstrument		指定交易合约指针
YDAccount		填写NULL，表示使用当前API登录账户

现货报单的参数说明如下：

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Normal
	OrderRef	客户报单编号，柜台在回报中会带回客户报单编号，投资者可以以此将报单和回报对应起来。
	Direction	YD_D_Buy：买 YD_D_Sell：卖
	OffsetFlag	Direction为YD_D_Buy时，填YD_OF_Open Direction为YD_D_Sell时，填YD_OF_Close
	HedgeFlag	填YD_HF_Speculation
	OrderType	YD_ODT_Limit：限价单 YD_ODT_Market：市价单 YD_ODT_FAK：FAK单 YD_ODT_FOK：FOK单
	Price	填写价格，价格不能超过涨跌停板限制，易达不控制价格波动带
	OrderVolume	买卖数量，受到以下限制： 不能超过合约的MaxMarketOrderVolume、MaxLimitOrderVolume 不能低于合约的MinMarketOrderVolume、MinLimitOrderVolume 不能超过持仓量限制 1买卖数量始终表示1股股票、1份基金和1张债券

参数	字段	说明
	OrderTriggerType	填YD_OTT_NoTrigger
	TriggerPrice	填0
	ExchangeOrderAttribute	填0
YDInstrument		指定交易合约指针
YDAccount		填写NULL，表示使用当前API登录账户

> **Tip**
>
> 在现货交易中，股票以股为单位，基金以份为单位，一买卖数量股票表示一股股票，一买卖数量基金表示一份基金。对应债券，深交所以张为单位，一买卖数量债券表示一张债券，但上交所以十张为单位，一买卖数量债券表示十张债券。
>
> 为提供统一的使用体验，易达柜台中交易的沪深交易所债券均以张为单位，即一买卖数量债券表示一张债券，由易达柜台负责与交易所和其他系统间的数量关系转换。

其中，易达系统中定义的指令类型OrderType是交易所指令类型条件的组合，针对每个交易所选择最贴近易达指令类型的组合进行设置，以下为各个交易所中的对照关系：

交易所	YD_ODT_Limit	YD_ODT_FAK	YD_ODT_Market	YD_ODT_FOK
CFFEX	FFEX_FTDC_TC_GFD FFEX_FTDC_OPT_LimitPrice FFEX_FTDC_VC_AV	FFEX_FTDC_TC_IOC FFEX_FTDC_OPT_LimitPrice FFEX_FTDC_VC_AV	FFEX_FTDC_TC_IOC FFEX_FTDC_OPT_AnyPrice FFEX_FTDC_VC_AV	FFEX_FTDC_TC_IOC FFEX_FTDC_OPT_LimitPrice FFEX_FTDC_VC_CV
SHFE	SHFE_FTDC_TC_GFD SHFE_FTDC_OPT_LimitPrice SHFE_FTDC_VC_AV	SHFE_FTDC_TC_IOC SHFE_FTDC_OPT_LimitPrice SHFE_FTDC_VC_AV	不支持，若使用则转换为： SHFE_FTDC_TC_IOC SHFE_FTDC_OPT_AnyPrice SHFE_FTDC_VC_AV	SHFE_FTDC_TC_IOC SHFE_FTDC_OPT_LimitPrice SHFE_FTDC_VC_CV
DCE	OT_LO OA_NONE	OT_LO OA_FAK	OT_MO OA_FAK	OT_LO OA_FOK

CZCE	FID_OrderType=0 //限价
	FID_MatchCondition=3 //当日有效
		FID_OrderType=0 //限价
		FID_MatchCondition=2 //即时部分成交
			FID_OrderType=1 //市价
			FID_MatchCondition=2 //即时部分成交
				FID_OrderType=0 //限价
				FID_MatchCondition=1 //即时全部成交

GFEX	OT_LO OA_NONE	OT_LO OA_FAK	OT_MO OA_FAK	OT_LO OA_FOK
SSE股票期权平台	OrderType=Limit TimeInForce=GFD	不支持，若使用则转换为 YD_ODT_FOK	OrderType=Market TimeInForce=IOC	OrderType=Limit TimeInForce=FOK
SSE竞价平台	OrderType=Limit	不支持	OrderType=Market	不支持
SSE债券平台	OrderType=Limit	不支持	不支持	不支持
SZSE股票期权	OrderType=Limit TimeInForce=GFD MinQty=0	不支持，若使用则转换为 YD_ODT_FOK	OrderType=Market TimeInForce=IOC MinQty=0	OrderType=Limit TimeInForce=GFD MinQty=OrderVolume
SZSE现货	OrderType=Limit TimeInForce=GFD MinQty=0	不支持	OrderType=Market TimeInForce=IOC MinQty=0	不支持

7.1.2 批量报单

批量报单可以一次性报不超过16个报单，可以满足客户多腿同时报单的需求。与多次调用普通报单相比，批量报单的整体性能并不占优，不建议出于性能目的使用批量报单。下面将从发送和接收两方面进行分析：

- 客户端发送时，批量报单的优点是可以减少API调用次数，但是需要把所有报单全部准备好以后才能进行发送，而普通报单却可以准备好一张立即发送一张，对于资源的利用率更高；而且批量报单在一个大包里面发出，发送时间更长
- 柜台接收时，批量报单也是顺序接收和处理的，这与多个普通报单的处理性能是基本相同的

```
virtual bool insertMultiOrders(unsigned count,YDInputOrder inputOrders[],const YDInstrument
*instruments[],const YDAccount *pAccount=NULL)
```

批量报单全部发送成功返回true，部分或全部发送失败返回false。参数填写方法如下：

参数	说明
unsigned count	报单张数，最多不能超过16张
YDInputOrder inputOrders[]	报单数组，与合约指针数组一一对应
YDInstrument *instruments[]	交易合约指针数组
YDAccount	填写NULL，表示使用当前API登录账户

易达柜台收到批量报单后，对每张报单的处理方式与普通报单的单张报单的处理方式是完全相同的：即每张报单都是独立事务，已经成功处理的单子不会因为后续单子风控失败而整体回退，同时排在前面的错单亦不会影响后面报单的处理。因此，批量报单的回报、撤单都与普通报单的相同，只不过会有多次回报回调。

7.1.3 本地风控报单

在ydExtendedApi中，增加了一系列在客户端风控并报单的指令，这些指令在相对应的报单指令上增加了本地钱仓和风控检查，如果没有通过风控，则会直接返回false，错误原因查看YDInputOrder中的ErrorNo获取。相比原始报单指令，当指令较容易被柜台风控拦截时，本地风控报单指令可以更快的知道报单是否可以**大概率**通过柜台的钱仓和风控检查，而节约了与柜台往返的交互时间，相对应的，也会增加客户端报单的时间开销。因此，请根据实际情况选择使用相应的报单指令。

```
virtual bool checkAndInsertOrder(YDInputOrder *pInputOrder,const YDInstrument
*pInstrument,const YDAccount *pAccount=NULL)
```

使用本地风控报单时，YDInputOrder中的OrderRef是由API自行从1开始编码，即便用户给OrderRef赋值，也会被API覆盖掉。该OrderRef编码机制利用了ydExtendedApi中提供的连接号编码机制，即本地风控报单天然支持多连接号。具体内容请参考多连接。在调用函数返回后，可以在传入的YDInputOrder.OrderRef中找到本次报单使用的OrderRef，如果检查失败的，同样可以在YDInputOrder.ErrorNo中找到报单风控失败的原因。

通过本地风控检查的报单不代表一定能通过柜台的风控检查，可能但不限于以下原因：

- 行情波动较大，柜台和客户端刷新持仓盈亏和保证金的时机不一致，导致客户端资金可能足够但柜台不足的情况
- 投资者使用多个策略在同一个账户内交易，可能存在因为其他策略报单的影响而导致柜台检查失败的情况
- 部分风控措施只在柜台端检查

使用本地风控的批量报单指令时，只有全部通过风控检查才能报送，这与批量报单在柜台端的处理行为是不同的。

投资者也可以使用以下指令只检查而不报单：

```
virtual bool checkOrder(YDInputOrder *pInputOrder,const YDInstrument *pInstrument,const
YDAccount *pAccount=NULL)
```

7.1.4 报单回报

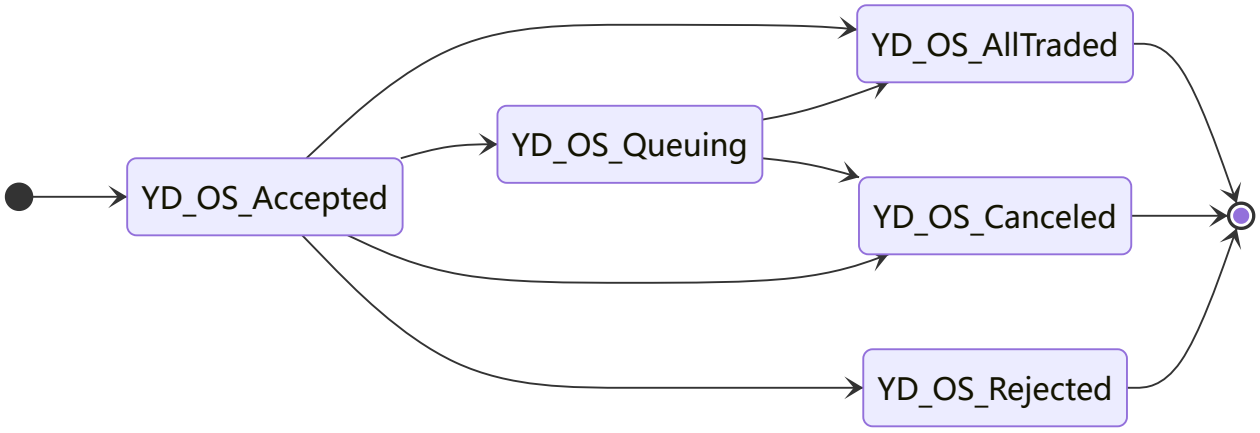
所有正确或者错误的交易所回报都将通过notifyOrder返回。收到回报后，若YDOrder.ErrorNo为0，则按照正确的报单回报处理；若YDOrder.ErrorNo为非0，则按照错误的报单回报处理。

```
virtual void notifyOrder(const YDOrder *pOrder,const YDInstrument *pInstrument,const YDAccount
*pAccount)
```

若报单通过柜台风控检查并向交易所发送后，在收到交易所回报前，默认情况下是不会向投资者发送回报的。但是投资者可以请经纪商打开柜台回报功能以收取柜台回报，可以通过查看YDAccount中AccountFlag是否设置了YD_AF_NotifyOrderAccept来判断柜台回报是否开通成功。柜台回报可以作为柜台收到投资者报单申请的凭据，担心UDP报单丢失的投资者可使用本机制更早发现UDP丢包。柜台回报通过notifyOrder返回，其OrderStatus为YD_OS_Accepted，除

了OrderSysID和InsertTime未赋值外，其他字段均已正确赋值，可以正常获取使用。除询价和大商所、广期所的期权对冲指令外，其他使用insertOrder系列报单指令报单的都会收到柜台回报。

每当收到交易所回报，易达柜台会据此的向投资者通过notifyOrder发送报单回报。回报的状态、数量和顺序与交易所具体行为有关，例如某些交易所FAK单、FOK单不会返回YD_OS_Queuing状态而就只有一个终态报单回报，再例如中金所的限价单如果在首次进入撮合队列时就全部成交的话那只会有一个全部成交的回报而不会有队列中的回报。交易所没有也不会承诺报单回报的行为，且有权随时改变该行为。但无论是易达还是交易所都将确保回报状态永远是按照报单状态机流转的，因此，投资者在开发策略程序时请不要依赖于交易所的回报行为，而是要做到无论以收到何种报单回报顺序都能正确处理。报单的状态机图如下所示。



notifyOrder的返回的参数说明如下，从YDInputOrder中带回的字段不再赘述：

参数	字段	说明
YDOrder	YDOrderFlag	固定为YD_YOF_Normal
	TradeVolume	报单的累计成交量
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	OrderSysID	对于期货交易所，为交易所返回的系统报单号 对于现货交易所，为经过转换的虚拟交易所报单编号，要获取真实的交易所报单编号请参见下文 若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	对于期货交易所，为交易所返回的系统报单号 对于现货交易所，为经过转换的虚拟交易所报单编号，要获取真实的交易所报单编号请参见下文 不会截取交易所报单编号，保留全精度
	InsertTime	从该交易日开始（17点整）到报单时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String

参数	字段	说明
	InsertTimeStamp	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	CancelTimeStamp	从该交易日开始（17点整）到撤单时间的豪秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	OrderTriggerStatus	报单触发状态 YD_OTS_NotTriggered：未触发 YD_OTS_Triggered：已触发
	OrderStatus	可能的状态如下，回报状态机见下图： YD_OS_Accepted：柜台接受，只会出现在柜台回报中 YD_OS_Queuing：队列中 YD_OS_AllTraded：全部成交 YD_OS_Canceled：撤单，包含部成部撤 YD_OS_Rejected：交易所错单
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		报单合约指针
YDAccount		当前API登录账户

由于上交所、深交所使用的交易所报单编号、组合编号、行权编号并不是整数，易达无法直接把交易所返回的这些编号直接放到OrderSysID字段中，而是会把交易所原生编号经过数字化转换后填入OrderSysID，若想查询交易所原生编号，可以使用以下两种方法。

对使用ydApi和ydExtendedApi的投资者，可以通过YDListener的notifyIDFromExchange回调函数收到原生报单编号，只有当发生交易所编号发生转换才会收到此类回调，在期货交易所交易时不会收到此回调。

```
virtual void notifyIDFromExchange(const YDIDFromExchange *pIDFromExchange,const YDExchange *pExchange)
```

返回的参数说明如下：

参数	字段	说明
YDIDFromExchange	IDType	ID类型，可能的值如下： YD_IDT_NormalOrderSysID：普通报单编号 YD_IDT_QuoteDerivedOrderSysID：报价衍生报单编号 YD_IDT_OptionExecuteOrderSysID：期权行权报单编号 YD_IDT_OptionAbandonExecuteOrderSysID：期权放弃行权报单编号 YD_IDT_RequestForQuoteOrderSysID：询价报单编号 YD_IDT_CombPositionOrderSysID：组合报单编号 YD_IDT_OptionExecuteTogether：合并行权报单编号 YD_IDT_Mark：组合报单编号 YD_IDT_OptionSelfClose：期权对冲编号 YD_IDT_FreezeUnderlying：现券锁定与解锁编号 YD_IDT_Cover：备兑报单编号 YD_IDT_TradeID：成交编号 YD_IDT_CombPositionDetailID：组合明细编号 YD_IDT_QuoteSysID：报价编号
	IDInSystem	经过易达柜台转换的ID值，可能会被截断
	LongIDInSystem	经过易达柜台转换的全精度ID值
	IDFromExchange	交易所原生的ID值，最大长度24字节
YDExchange		交易所指针

对使用ydExtendedApi的投资者，还可以直接查询原生编号，查询时需要输入的参数的含义可以参见上表，返回的原生ID最大长度为24字节。

```
virtual const char *getIDFromExchange(const YDExchange *pExchange,int idType,int idInSystem)
virtual const char *getLongIDFromExchange(const YDExchange *pExchange,int idType,long long longIdInSystem)
```

7.1.5 扩展报单回报

若投资者使用了YDExtendedListener，那么可以通过notifyExtendedOrder在以下情况下会收到回报：

- notifyOrder回调时，无论是正确报单还是错误报单，无论柜台拒单还是交易所拒单
- 当YDExtendedOrder上的属性有任何变化时
- 当使用checkAndInsertOrder报单且发送成功后

```
virtual void notifyExtendedOrder(const YDExtendedOrder *pOrder)
```

由于YDExtendedOrder是继承自YDOrder，下面仅列举YDExtendedOrder的专属字段：

字段	说明
m_pInstrument	报单合约指针
m_pCombPositionDef	报单类型为YD_YOF_CombPosition组合或者解组时有效，为传统组合持仓定义指针
m_pAccount	报单所属的投资者指针
m_pInstrument2	报单类型为YD_YOF_OptionExecuteTogether合并行权时有效，为合并行权的第二个合约的指针

7.1.6 成交回报

每当有成交产生就会通过notifyTrade返回成交回报，成交回报中的OrderRef、OrderLocalID和OrderSysID均可用于关联到具体报单。

```
virtual void notifyTrade(const YDTrade *pTrade,const YDInstrument *pInstrument,const YDAccount *pAccount)   {}
```

成交回报的返回值如下所示，与报单相同的字段不再重复：

参数	字段	说明
YDTrade	TradeID	交易所返回的成交编号 若交易所报单编号超过TradeID的最大长度，则会截取部分
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	LongTradeID	全精度的交易所返回的成交编号
	OrderSysID	对于期货交易所，为交易所返回的系统报单号 对于现货交易所，为经过转换的虚拟交易所报单编号，要获取真实的交易所报单编号请参见下文 若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	对于期货交易所，为交易所返回的系统报单号 对于现货交易所，为经过转换的虚拟交易所报单编号，要获取真实的交易所报单编号请参见下文 不会截取交易所报单编号，保留全精度
	OrderRef	投资者报单编号
	OrderGroupID	报单所属报单组编号
	Price	成交价格
	Volume	成交数量
	Commission	成交手续费
	TradeTime	从该交易日开始（17点整）到成交时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	TradeTimeStamp	从该交易日开始（17点整）到成交时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
YDInstrument		成交合约指针
YDAccount		当前API登录账户

7.1.6.1 扩展成交回报

若投资者使用了YDExtendedListener，那么在任何notifyTrade回调的时候都会同时回调notifyExtendedTrade：

```
virtual void notifyExtendedTrade(const YDExtendedTrade *pTrade)
```

由于YDExtendedTrade是继承自YDTrade，下面仅列举YDExtendedTrade的专属字段：

字段	说明
m_pInstrument	成交合约指针
m_pAccount	成交所属的投资者指针

7.2 撤单

投资者可以调用下列函数撤回处于队列中状态（OrderStatus为YD_OS_Queuing）的报单。

```
virtual bool cancelOrder(YDCancelOrder *pCancelOrder,const YDExchange *pExchange,const YDAccount *pAccount=NULL)
virtual bool cancelMultiOrders(unsigned count,YDCancelOrder cancelOrders[],const YDExchange *exchanges[],const YDAccount *pAccount=NULL)
```

7.2.1 普通撤单

可以使用下列方法撤回单笔报单，如果该函数返回false，则说明到柜台的网络发生了中断。

```
virtual bool cancelOrder(YDCancelOrder *pCancelOrder,const YDExchange *pExchange,const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	字段	说明
YDCancelOrder	YDOrderFlag	填写待撤单的报单的报单类型
	OrderSysID	交易所报单编号
	LongOrderSysID	全精度的交易所报单编号
	OrderGroupID	指定报单和报价所属的逻辑组，与OrderRef配合使用可实现OrderRef撤单
	OrderRef	投资者报单编号。与OrderGroupID配合使用可实现OrderRef撤单
YDExchange		撤单所在交易所
YDAccount		填写NULL，表示使用当前API登录账户

柜台支持三种撤单方式，包括LongOrderSysID、OrderSysID和OrderRef。OrderRef撤单方式允许投资者收到回报前撤单，目前支持中金所、上期所、能源所、大商所、上交所、深交所。

> 广期所撤单必须使用交易所返回的信息，因此无法支持在收到回报前的撤单。如果投资者在柜台收到交易所回报前向广期所柜台发送OrderRef撤单请求，会被柜台拒绝并返回YD_ERROR_ExchangeConnectionSendError=80错误；如果投资者在柜台收到交易所回报后向广期所柜台发送OrderRef撤单请求，柜台会使用回报信息向交易所发起撤单请求。因此，广期所的OrderRef撤单存在不确定性，不推荐使用。

易达柜台收到撤单指令后，将按照如下逻辑决定使用何种撤单方式：

- 首先判断OrderGroupID。若OrderGroupID为0，则继续判断LongOrderSysID的值，否则转到后续OrderGroupID非0的逻辑：
  - 若LongOrderSysID非0，则使用撤单指令中填写的LongOrderSysID撤单
  - 若LongOrderSysID为0，则使用撤单指令中填写的OrderSysID撤单
- 若OrderGroupID非0，则撤回由撤单指令中OrderGroupID和OrderRef唯一确定的报单

7.2.2 批量撤单

批量撤单可以一次性撤不超过16个报单，其特性与批量报单类似。

```
virtual bool cancelMultiOrders(unsigned count,YDCancelOrder cancelOrders[],const YDExchange *exchanges[],const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	说明
unsigned count	撤单张数，最多不能超过16张
YDCancelOrder cancelOrders[]	撤单信息数组，与交易所指针数组一一对应
YDExchange *exchanges[]	对应于每个撤单信息所在的交易所指针数组
YDAccount	填写NULL，表示使用当前API登录账户

7.2.3 撤单回报

如果撤单成功，那么被撤报单会通过notifyOrder返回其最新状态，具体请参阅报单回报。

如果撤单失败，无论是柜台拒绝撤单还是交易所拒绝撤单，都会通过如下回调返回。请检查YDFailedCancelOrder中的ErrorNo获取撤单失败的原因，通常是该报单已经完成成交或者已撤单的错误。**请注意，易达无法保证撤单失败回报与投资者发送的撤单指令在数量上和指令内容上的严格对应，强烈建议投资者仅将撤单失败回报用于记录日志。交易策略不应依赖于撤单失败回报，而应在撤单指令发出后建立超时机制，若在设定时间内没有收到对应报单状态变化的回报，则重新发送撤单指令。**

```
virtual void notifyFailedCancelOrder(const YDFailedCancelOrder *pFailedCancelOrder,const YDExchange *pExchange,const YDAccount *pAccount)
```

上述方法的参数说明如下：

参数	字段	说明
YDCancelOrder	AccountRef	账户序号
	ExchangeRef	交易所序号
	YDOrderFlag	错误撤单对应报单的报单类型
	OrderSysID	交易所报单编号
	LongOrderSysID	全精度的交易所报单编号
	OrderGroupID	报单逻辑组ID
	OrderRef	投资者报单编号
	ErrorNo	错误代码

参数	字段	说明
	IsQuote	是否撤报价回报
YDExchange		错误撤单所在交易所
YDAccount		错误撤单账户

一般情况下，撤单失败回报返回的信息与撤单使用的方法相对应，即：

- 使用OrderSysID或者LongOrderSysID撤单的回报中填写OrderSysID和LongOrderSysID，OrderGroupID和OrderRef无意义
- 使用OrderGroupID与OrderRef撤单的回报中填写OrderGroupID和OrderRef，OrderSysID和LongOrderSysID无意义

但以下情况下，使用OrderGroupID与OrderRef撤单的撤单失败回报与上述情况不同：

- 由交易所返回的撤单失败回报将不会给投资者转发
- 报单回报已返回，且撤单时遇到流控或席位断线等错误时，撤单失败回报会填写OrderSysID和LongOrderSysID

因此，当收到撤单失败回报时，需判断其返回信息的方式。若OrderGroupID为0，则通过OrderSysID或LongOrderSysID找到对应的报单；若OrderGroupID为非0，则通过OrderGroupID和OrderRef找到对应的报单。

通常投资者会使用OrderRef建立报单索引，当收到填写了OrderSysID和LongOrderSysID的错误撤单回报时，可以使用以下方法获取对应的OrderRef和OrderGroupID：

```
virtual const YDExtendedOrder *getOrder(int orderSysID,const YDExchange *pExchange,int YDOrderFlag=YD_YOF_Normal)
virtual const YDExtendedOrder *getOrder(long long longOrderSysID,const YDExchange *pExchange,int YDOrderFlag=YD_YOF_Normal)
```

7.3 备兑业务

7.3.1 现货锁定与解锁

上交所股票期权备兑开仓和普通转备兑前，需要首先锁定现货仓位。深交所执行备兑业务时其交易系统内部执行了冻结，因此可以直接开展备兑业务，无需事先冻结。

请按照下列参数说明，使用insertOrder指令向交易所发起锁定和解锁操作。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_FreezeUnderlying，说明该报单是现货锁定业务
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	Direction	YD_D_Freeze：锁定 YD_D_Unfreeze：解锁
	OrderVolume	填写锁定和解锁的数量
	HedgeFlag	填写YD_HF_Covered
YDInstrument		指定待锁定或解锁的现货合约指针
YDAccount		填写NULL，表示使用当前API登录账户

7.3.2 备兑开平仓

请按照下列参数说明，使用insertOrder指令向交易所发起备兑开仓和平仓操作。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Normal
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	Direction	YD_D_Buy：买 YD_D_Sell：卖
	OffsetFlag	YD_OF_Open：开仓 YD_OF_Close：平仓
	HedgeFlag	填写YD_HF_Covered
	OrderType	可以根据需要选择填写以下值： YD_ODT_Limit：限价单 YD_ODT_Market：市价单 YD_ODT_FOK：FOK单
	Price	填写价格
	OrderVolume	开平仓数量
YDInstrument		指定备兑开平仓的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

7.3.3 普通与备兑转换

请按照下列参数说明，使用insertOrder指令向交易所发起普通转备兑或者备兑转普通的操作。请注意，上交所和深交所均支持普通转备兑业务，但是只有深交所支持备兑转普通业务。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Cover
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	Direction	YD_D_Normal2Covered：普通转备兑 YD_D_Covered2Normal：备兑转普通，仅适用于深交所
	HedgeFlag	填写YD_HF_Covered
	OrderVolume	转换数量
YDInstrument		指定待转换的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

7.4 做市商

7.4.1 询价

对于上期所、能源所、中金所、大商所、郑商所、广期所，可参照下列参数说明，使用insertOrder发送询价请求。询价必须针对可以询价的合约，否则没有效果。

> 上期所、能源所将询价纳入信息量计算，为防止投资者信息量意外超出后被收取高额的手续费，易达提供了事前信息量风控和事后信息量风控。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_RequestForQuote
	Direction	填写YD_D_Buy
	OffsetFlag	填写YD_OF_Open
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 大商所和广期所并不使用该标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
	OrderType	填写YD_ODT_Limit
YDInstrument		指定待询价的合约指针
YDAccount		填写NULL，表示使用当前API登录账户

即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，询价指令也不会发送柜台回报。

为减少不必要的流量，询价回报notifyRequestForQuote只在做市商系统（柜台授权文件中含有MarketMaker标志）中通知，普通系统无法收到询价回报。

```
virtual void notifyRequestForQuote(const YDRequestForQuote *pRequestForQuote,const YDInstrument *pInstrument)
```

以下为notifyRequestForQuote中返回信息的说明。

参数	字段	说明
YDRequestForQuote	RequestTime	从该交易日开始（17点整）到询价时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	RequestForQuoteID	询价编号。若返回长度过长，则会被截断
	LongRequestForQuoteID	全精度的询价编号
YDInstrument		询价合约指针

在ydExtendedApi中，易达提供了主动查询询价号的方法，使用方法如下：

```
virtual const YDExtendedRequestForQuote *getRequestForQuote(const YDInstrument *pInstrument)
```

上述方法的返回值YDExtendedRequestForQuote是YDRequestForQuote的子类，因此可以直接读取询价时间RequestTime和RequestForQuoteID询价编号。若查询的查阅不存在询价，那么上述方法将返回NULL。

7.4.2 报价

易达支持上期所、能源所、中金所、大商所、广期所、郑商所、上交所、深交所的做市商报价业务，报价指令只能在打开了做市商功能的易达柜台中使用（柜台授权文件中含有MarketMaker标志），可以通过查看SystemParam中的MarketMaker标志查看当前柜台是否支持报价功能，详情参见系统参数。

7.4.2.1 普通报价

做市商可以使用下列方法发送单张报价单，如果该函数返回false，则说明到柜台的网络发生了中断。

```
virtual bool insertQuote(YDInputQuote *pInputQuote,const YDInstrument *pInstrument,const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	字段	说明
YDInputQuote	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	BidOffsetFlag	买开平标志
	BidHedgeFlag	买投保标志 中金所：交易所接口实际只支持两侧投保标志相同，易达不进行检查，但是报入交易所时使用买入方的投保标志对应的客户号 其他交易所：买卖方投保标志可以不同
	AskOffsetFlag	卖开平标志
	AskHedgeFlag	卖投保标志
	BidPrice	买价格，买价必须低于卖价
	AskPrice	卖价格
	BidVolume	买数量。根据YDExchange.QuoteVolumeRestriction有不同的买卖数量限制： 0（大商所、广期所、上交所、深交所）：BidVolume和AskVolume至少一个大于0，另一个大于等于0 1（中金所）：BidVolume和AskVolume必须大于0，但是可以不等 2（上期所、能源所、郑商所）：BidVolume和AskVolume必须相等且大于0
	AskVolume	卖数量
	OrderRef	客户报单编号
	YDQuoteFlag	报价标志，该字段是位图，可以同时设置多个标志 YD_YQF_ResponseOfRFQ：自动填写询价号表示应价，支持的有上期所、能源所、大商所、广期所、郑商所，其他交易所无询价号 YD_YQF_ReplaceLastQuote：是否顶最后发送的报价，只对中金所生效 YD_YQF_ReplceSpecifiedQuote：是否顶指定报价，只对中金所生效

参数	字段	说明
	ExchangeQuoteAttribute	交易所报价属性，由报价所属交易所解释其含义。 YD_EOA_DCE_GIS=1：大商所GIS当前小节有效报单标志
	ReplaceLongQuoteSysID	中金所顶指定报价时，需要指被顶单的QuoteSysID
YDInstrument		指定待报价的合约指针
YDAccount		填写NULL，表示使用当前API登录账户

在发起报价请求后，为了能统一所有交易所的报价业务，易达柜台会为该报价生成对应的报单，这些报单的YDOrderFlag为YD_YOF_QuoteDerived，如果是双边报价会产生两笔衍生报单，如果是单边报价则只会产生一笔衍生报单。衍生的报单只存在于易达内部，并不会发送到交易所，报价单的成交、报单状态将会反映在衍生报单上面，而报价单自身是没有状态的。衍生报单YDOrder的字段与报价YDQuote字段的关系如下所示：

- 买卖衍生报单的Direction分别对应YD_D_Buy和YD_D_Sell
- 买卖衍生报单的OffsetFlag分别对应BidOffsetFlag和AskOffsetFlag
- 买卖衍生报单的HedgeFlag分别对应BidHedgeFlag和AskHedgeFlag
- 买卖衍生报单的Price分别对应BidPrice和AskPrice
- 买卖衍生报单的OrderVolume分别对应BidVolume和AskVolume
- 买卖衍生报单的OrderRef都为报价单的OrderRef
- 买卖衍生报单的YDOrderFlag都为YD_YOF_QuoteDerived
- 买衍生报单的OrderLocalID按照普通报单顺序编号，卖衍生报单OrderLocalID一定是买衍生报单OrderLocalID+1
- 买卖衍生报单的OrderSysID分别对应BidOrderSysID和AskOrderSysID

7.4.2.2 批量报价

批量报价可以一次性报不超过12个报价，其特性与批量报单类似。

```
virtual bool insertMultiQuotes(unsigned count,YDInputQuote inputQuotes[],const YDInstrument *instruments[],const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	说明
unsigned count	报价单张数，最多不能超过12张
YDInputQuote inputQuotes[]	报单数组，与合约指针数组一一对应
const YDInstrument *instruments[]	对应于每个报价的交易所指针数组
YDAccount	填写NULL，表示使用当前API登录账户

7.4.2.3 报价回报

所有正确或者错误的交易所报价回报会通过notifyQuote返回。收到回报后，若YDQuote.ErrorNo为0，则按照正确的报价回报处理；若YDQuote.ErrorNo为非0，则按照错误的报价回报处理。

若报价失败，则只会收到notifyQuote；若报价成功，则会通过以下回调函数收到通知，其中：

- notifyQuote：每个报价单只有一个回报，表示交易所已经收到了报价请求，不保证与notifyOrder之间的次序
- notifyOrder：每次衍生报单状态发生变化就会收到该回调，其状态机同普通报单

- notifyTrade：同普通报单的报价，每当报价有成交产生就会有成交回报，可以通过OrderLocalID关联到衍生报单，可以通过OrderSysID关联到报价单

```
virtual void notifyQuote(const YDQuote *pQuote,const YDInstrument *pInstrument,const YDAccount *pAccount) {}
virtual void notifyOrder(const YDOrder *pOrder,const YDInstrument *pInstrument,const YDAccount *pAccount)   {}
virtual void notifyTrade(const YDTrade *pTrade,const YDInstrument *pInstrument,const YDAccount *pAccount)   {}
```

与报单回报类似，notifyQuote和notifyOrder的顺序与交易所具体行为有关，例如，在收到双边报价后，中金所先返回报价的notifyQuote后返回两张衍生报单的notifyOrder，而上期所先返回两张衍生报单的notifyOrder后返回报价的notifyQuote。交易所没有也不会承诺报价回报的行为，且有权随时改变该行为，因此，投资者在开发策略程序时请不要依赖于交易所的回报行为，而是要做到无论以收到何种回报顺序都能正确处理。

其中，notifyOrder和notifyTrade同普通报单，notifyQuote中的YDQuote是YDInputQuote的子类，除了常规的RealConnectionID和ErrorNo，YDQuote新增信息如下：

参数	字段	说明
YDQuote	QuoteSysID	当ErrorNo为YD_ERROR_InvalidGroupOrderRef时，表示当前柜台已收到的最大OrderRef；否则表示报价编号，用于撤报价。若交易所返回值超长，则会被截断
	BidOrderSysID	买报价的OrderSysID，卖单边报价时为0。若交易所返回值超长，则会被截断
	AskOrderSysID	卖报价的OrderSysID，买单边报价时为0。若交易所返回值超长，则会被截断
	RequestForQuoteID	当YDQuoteFlag为YD_YQF_ResponseOfRFQ时，记录了报单响应的询价号，否则为0。若交易所返回值超长，则会被截断
	LongQuoteSysID	全精度的报价编号，用于撤报价
	LongBidOrderSysID	全精度的买报价的OrderSysID，卖单边报价时为0
	LongAskOrderSysID	全精度的卖报价的OrderSysID，买单边报价时为0
	LongRequestForQuoteID	全精度的当YDQuoteFlag为YD_YQF_ResponseOfRFQ时，记录了全精度的报单响应的询价号，否则为0
YDInstrument		报价合约指针
YDAccount		当前API登录账户指针

7.4.2.4 扩展报价回报

若投资者使用了YDExtendedListener，那么可以通过notifyExtendedQuote在以下情况下会收到回报：

- notifyQuote回调时，无论是正确报价还是错误报价，无论柜台拒单还是交易所拒单
- 当YDExtendedQuote上的属性有任何变化时
- 当使用checkAndInsertQuote报单且发送成功后

```
virtual void notifyExtendedQuote(const YDExtendedQuote *pQuote)
```

由于YDExtendedQuote是继承自YDQuote，下面仅列举YDExtendedQuote的专属字段：

字段	说明
BidOrderFinished	买腿对应的衍生报单是否已经完成 当衍生报单是撤单、全成、拒绝时被认为是完成 若报价买量为0，则直接被认为是完成
AskOrderFinished	卖腿对应的衍生报单是否已经完成 当衍生报单是撤单、全成、拒绝时被认为是完成 若报价卖量为0，则直接被认为是完成
m_pInstrument	报价合约指针
m_pAccount	报价所属的投资者指针

7.4.3 撤报价

投资者可以调用下列方法撤回未完成的报价。

```
virtual bool cancelQuote(YDCancelQuote *pCancelQuote,const YDExchange *pExchange,const YDAccount *pAccount=NULL)
virtual bool cancelMultiQuotes(unsigned count,YDCancelQuote cancelQuotes[],const YDExchange *exchanges[],const YDAccount *pAccount=NULL)
```

> 基于目前大商所的实现，若撤报价时一腿的衍生报单为全部成交、已撤单，另一腿为队列中时，交易所会先返回撤单错误给柜台，柜台通过notifyFailedCancelQuote返回撤单错误，然后再返回原队列中的腿已撤单的回报，柜台会通过notifyOrder返回被撤单的衍生报单的报单回报。请投资者注意规避该问题。

7.4.3.1 普通撤报价

可以使用下列方法撤回单笔报价，如果该函数返回false，则说明到柜台的网络发生了中断。

```
virtual bool cancelQuote(YDCancelQuote *pCancelQuote,const YDExchange *pExchange,const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	字段	说明
YDCancelQuote	QuoteSysID	交易所报价编号
	LongQuoteSysID	全精度的交易所报价编号
	OrderGroupID	指定报价所属的逻辑组，与OrderRef配合使用可实现OrderRef撤报价
	OrderRef	与OrderGroupID配合使用可实现OrderRef撤报价
YDExchange		待撤回报价的交易所指针
YDAccount		填写NULL，表示使用当前API登录账户

柜台支持三种撤报价方式，包括LongQuoteSysID、QuoteSysID和OrderRef。OrderRef撤单方式允许投资者收到回报前撤报价，目前支持中金所、上期所、能源所、大商所的单边报价。

> 广期所撤报价（包括单边报价和双边报价）和大商所撤双边报价必须使用交易所返回的信息，因此无法支持在收到回报前的撤报价。如果投资者在柜台收到交易所回报前向广期所或大商所柜台发送OrderRef撤报价请求，会被柜台拒绝并返回YD_ERROR_ExchangeConnectionSendError=80错误；如果投资者在柜台收到交易所回报后向广期所或大商所柜台发送OrderRef撤单请求，柜台会使用回报信息向交易所发起撤单请求。因此，广期所撤报价（包括单边报价和双边报价）和大商所撤双边报价的OrderRef撤报价存在不确定性，不推荐使用。

易达柜台收到撤单指令后，将按照如下逻辑决定使用何种撤单方式：

- 首先判断OrderGroupID。若OrderGroupID为0，则继续判断LongQuoteSysID的值，否则转到后续OrderGroupID非0的逻辑：
  - 若LongQuoteSysID非0，则使用撤报价指令中填写的LongQuoteSysID撤报价
  - 若LongQuoteSysIDID为0，则使用撤报价指令中填写的OuoteSysID撤报价
- 若OrderGroupID非0，则撤回由撤报价指令中OrderGroupID和OrderRef唯一确定的报价

7.4.3.2 批量撤报价

批量撤报价可以一次性撤不超过16个报价，其特性与批量报单类似。

```
virtual bool cancelMultiQuotes(unsigned count,YDCancelQuote cancelQuotes[],const YDExchange *exchanges[],const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	说明
unsigned count	撤报价张数，最多不能超过16张
YDCancelQuote cancelQuotes[]	撤报价信息数组，与交易所指针数组一一对应
YDExchange *exchanges[]	对应于每个撤报价信息所在的交易所指针数组
YDAccount	填写NULL，表示使用当前API登录账户

7.4.3.3 撤衍生报单

大商所和广期所支持撤回衍生报单的方式撤回双边报价的一边，中金所、上期所、能源所、郑商所、上交所、深交所不支持。撤回衍生报单的方式同普通报单，请参考撤单的相应内容。与撤回报价类似，撤回一腿衍生报单时只会有报单回调notifyOrder，而没有报价回调notifyQuote，只有当两腿衍生报单都被撤回时，才会返回报价回调。

7.4.3.4 合约撤报价

与期货交易所不同，上交所和深交所需使用下列方法按合约撤回报价，如果该函数返回false，则说明到柜台的网络发生了中断。

```
virtual bool cancelQuoteByInstrument(YDCancelQuote *pCancelQuote,const YDInstrument *pInstrument,int cancelQuoteByInstrumentType,const YDAccount *pAccount=NULL)
```

上述方法的参数说明如下：

参数	字段	说明
YDCancelQuote		用于指定发单连接，其余与撤报价相关的业务字段请不要在结构体中填写
YDInstrument		待撤回报价的合约指针
cancelQuoteByInstrumentType		撤报价的方式，只对上交所有效，深交所只能两边同时撤： YD_CQIT_Buy=1：撤买边 YD_CQIT_Sell=2：撤卖边 YD_CQIT_Both=3：撤双边
YDAccount		填写NULL，表示使用当前API登录账户

7.4.3.5 撤报价回报

如果撤报价成功，则通过notifyOrder回调返回各个衍生报单的状态，请注意，撤回报价时**不会**有notifyQuote回调。

```
virtual void notifyOrder(const YDOrder *pOrder,const YDInstrument *pInstrument,const YDAccount *pAccount)   {}
```

如果撤报价失败，则通过notifyFailedCancelOrder回调返回错误信息，此时应检查YDQuote中的ErrorNo检查错误原因。**请注意，易达无法保证撤报价失败回报与投资者发送的撤报价指令在数量上和指令内容上的严格对应，强烈建议投资者仅将撤报价失败回报用于记录日志。交易策略不应依赖于撤报价失败回报，而应在撤报价指令发出后建立超时机制，若在设定时间内没有收到对应报价状态变化的回报，则重新发送撤报价指令。**

```
virtual void notifyFailedCancelQuote(const YDFailedCancelQuote *pFailedCancelQuote,const YDExchange *pExchange,const YDAccount *pAccount)
```

上述方法的参数说明如下：

参数	字段	说明
YDCancelQuote	AccountRef	账户序号
	ExchangeRef	交易所序号
	QuoteSysID	交易所报价编号
	LongQuoteSysID	全精度的交易所报价编号
	OrderGroupID	报单逻辑组ID
	OrderRef	投资者报单编号
	ErrorNo	错误代码
	IsQuote	是否错误撤报价回报
YDExchange		错误撤单所在交易所
YDAccount		错误撤单账户

一般情况下，撤报价失败回报返回的信息与撤报价使用的方法相对应，即：

- 使用QuoteSysID或者LongQuoteSysID撤报价的回报中填写QuoteSysID和LongQuoteSysID，OrderGroupID和OrderRef无意义
- 使用OrderGroupID与OrderRef撤报价的回报中填写OrderGroupID和OrderRef，QuoteSysID和LongQuoteSysID无意义

但以下情况下，使用OrderGroupID与OrderRef撤报价的撤报价失败回报与上述情况不同：

- 由交易所返回的撤报价失败回报将不会给投资者转发
- 报单回报已返回，且撤报价时遇到流控或席位断线等错误时，撤报价失败回报会填写QuoteSysID和LongQuoteSysID

因此，当收到撤报价失败回报时，需判断其返回信息的方式。若OrderGroupID为0，则通过QuoteSysID或LongQuoteSysID找到对应的报单；若OrderGroupID为非0，则通过OrderGroupID和OrderRef找到对应的报单。

通常投资者会使用OrderRef建立报单索引，当收到填写了QuoteSysID和LongQuoteSysID的错误撤报价回报时，可以使用以下方法获取对应的OrderRef和OrderGroupID：

```
virtual const YDExtendedQuote *getQuote(int quoteSysID,const YDExchange *pExchange)
virtual const YDExtendedQuote *getQuote(long long longQuoteSysID,const YDExchange *pExchange)
```

7.4.4 报价顶单

上期所、能源所、大商所、郑商所、中金所、上交所、深交所支持报价顶单业务（以报代撤），即可以通过新的报价顶替之前的相同合约的报价，可以减少撤单次数。

> 在CTP中，上期所和能源所平仓顶单时并不验仓。例如，某做市商有10手某合约持仓，在已有平8手的报价挂单的情况下，该做市商可以直接下平8手的报价单实现顶替前述报价单的效果。但是由于在其他交易所CTP并不支持平仓顶单不验仓功能，易达为上期所和能源所的平仓顶单特殊处理不利于做市商统一报单代码，因此易达在所有交易所均不支持超额平仓顶单功能。请注意，当挂单报价和顶单报价的平仓手数之和小于总持仓量时，顶单总是允许的，例如，某做市商有10手某合约持仓，在已有平4手的报价挂单的情况下，该做市商可以直接下平4手的报价单实现顶替前述报价单的效果。

投资者使用顶单业务报价时，可以使用普通报价的方法直接提交新的报价，回调情况等同于一次撤报单和一次报价产生的回报，因此请参考报价和撤报价相关内容。

中金所支持多层次报价，即可以通过在多个价位上报价，因此，中金所顶单支持不顶单、顶最后一单和顶指定单三种。报价时，如果YDQuote.YDQuoteFlag设置了YD_YQF_ReplaceLastQuote标志位，则表示顶最后发送报价；如果YDQuote.YDQuoteFlag设置了YD_YQF_ReplceSpecifiedQuote标志位，则表示顶指定报价，被顶报价需要在ReplaceLongQuoteSysID中指定；如果不设置，则表示不顶单，即会产生多层次报价。由于中金所多层次报价的特性，自成交判断会变得复杂，请务必参考中金所报价自成交检查。

上交所和深交所支持顶双边、顶单边两种。顶双边时，报价单中的买卖数量需大于0；顶单边时，被顶一边的数量需大于0，另一边的数量需等于0。顶单边报价时，交易所将遍历所有处于挂单状态的同合约报价单，根据被顶单类型和买卖方向逐一判断对被顶单的影响：

- 若被顶单是单边报价且买卖方向与顶单报价相同，则被顶报价被撤单
- 若被顶单是单边报价且买卖方向与顶单报价相反
    - 若价格交叉，则被顶报价被撤单
    - 若价格不交叉，则被顶报价保持不变
- 若被顶单是双边报价
    - 若与顶单报价买卖方向相同的一边已经成交，则被顶报价的这一边保持不变
    - 若与顶单报价买卖方向相同的一边仍然处于挂单，则被顶报价的这一边被撤单
    - 若与顶单报价买卖方向相反的一边已经成交，则被顶报价的这一边保持不变
    - 若与顶单报价买卖方向相反的一边仍然处于挂单
        - 若价格交叉，则被顶报价的这一边保持不变
        - 若价格不交叉，则被顶报价的这一边保持不变

7.5 组合与解组合持仓

在传统保证金模型中，大商所、广期所、上交所、深交所没有大边保证金业务，只能通过传统组合持仓优惠保证金；郑商所既有同合约单向大边保证金，也可以通过传统组合持仓优惠保证金。

- 对于大商所和广期所，结算时会进行自动传统组合持仓，因此开盘前这些持仓均处于组合状态。如果盘中有新开仓符合传统组合持仓定义且需要组合的，可以使用组合指令予以组合。平仓时，大商所和广期所组合自动解组，无需事先解组。
- 对于郑商所，结算时备兑会自动组合保证金，其他类型需要投资者申请，因此开盘前备兑和申请成功的持仓会处于组合状态，盘中时按照对锁（同合约大边）自动优惠保证金。平仓时，郑商所组合自动解组，无需事先解组。
- 对于上交所和深交所，结算时不会自动组合，只能在盘中通过组合指令进行组合。平仓时，上交所和深交所的持仓必须先解组，然后平仓，处于组合的持仓无法被平仓。

在组合保证金模型中，中金所支持发起组合和解组合持仓，其他交易所均不支持。

为了便于投资者查询当前柜台是否可以对某个组合定义发起组合或者解组合，易达提供了以下方法，其中YDCombPositionDef的获取方法参见传统组合持仓定义：

```
virtual bool canUseCombPositionDef(const YDCombPositionDef *pDef,const YDAccount *pAccount=NULL)
```

易达提供了3种不同的方法发送组合指令，分别是：

- 原生指令：ydApi和ydExtendedApi的用户可以调用insertCombPositionOrder发送原生组合和解组指令，ydExtendedApi的用户可以调用checkAndInsertCombPositionOrder发送原生组合和解组指令
- 自动指令：ydExtendedApi的用户可以调用autoCreateCombPosition发起自动组合指令
- 自动工具：运行自动组合工具ydAutoCP定时发送组合指令，该工具可以从这里下载或者向经纪商索取。

7.5.1 原生指令

ydApi和ydExtendedApi的用户可以调用insertCombPositionOrder发送原生组合和解组指令，ydExtendedApi的用户可以调用checkAndInsertCombPositionOrder发送原生组合和解组指令。与其他组合方法相比，原生指令提供了最大的自由度，投资者可以不受优先级约束随意选择需要组合的指令，而且原生指令也是唯一支持解组合的指令。同时，自动指令和自动工具都是基于原生指令开发的。

```
virtual bool insertCombPositionOrder(YDInputOrder *pInputOrder,const YDCombPositionDef *pCombPositionDef,const YDAccount *pAccount=NULL)
virtual bool checkAndInsertCombPositionOrder(YDInputOrder *pInputOrder,const YDCombPositionDef *pCombPositionDef,const YDAccount *pAccount=NULL)
```

checkAndInsertCombPositionOrder与insertCombPositionOrder的参数是相同的，区别在于checkAndInsertCombPositionOrder相对insertCombPositionOrder增加了API端对输入参数的合法性检查，如果合法性检查失败，则checkAndInsertCombPositionOrder会直接返回false，错误原因查看YDInputOrder中的ErrorNo获取，而不需要等待柜台检查后返回错误，这些合法性检查包括：

- 买卖方向、投保标志、组合数量的取值合法性检查
- 是否有足够的持仓进行组合，是否有足够的传统组合持仓进行解组

API本地合法性检查可以减少因字段不合法而被退回的时间，但是会在每个组合指令上增加额外的开销。好在组合指令并非是性能敏感型指令，因此推荐所有希望使用原生指令的ydExtendedApi用户都能使用checkAndInsertCombPositionOrder来发起组合指令。

insertCombPositionOrder和checkAndInsertCombPositionOrder的调用方法如下所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_CombPosition
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	Direction	YD_D_Make：组合 YD_D_Split：解组
	OrderType	填0
	HedgeFlag	填写YD_HF_Speculation
	OrderVolume	组合需要或者解组的数量。 组合时需要两腿的未传统组合持仓量都大于组合数量。 解组时需要传统组合持仓量大于解组数量。
	CombPositionDetailID	组合明细ID，只有当在上交所和深交所解组时需要填写，其他情况填0

参数	字段	说明
YDCombPositionDef		待组合的传统组合持仓定义指针，请参考传统组合持仓定义获取。
YDAccount		填写NULL，表示使用当前API登录账户

组合成功或者失败后，会通过以下回调接口通知投资者。

```
virtual void notifyCombPositionOrder(const YDOrder *pOrder,const YDCombPositionDef
*pCombPositionDef,const YDAccount *pAccount) {}
```

7.5.1.1 扩展组合回报

若投资者使用了YDExtendedListener，那么可以通过notifyExchangeCombPositionDetail在以下情况下会收到组合明细回报：

- 接收日初组合明细数据时，请参见日初传统组合持仓
- 当组合或者解锁指令成功导致组合明细发生变化时

```
virtual void notifyExchangeCombPositionDetail(const YDExtendedCombPositionDetail
*pCombPositionDetail)
```

YDExtendedCombPositionDetail的结构体请参见日初传统组合持仓。

7.5.2 自动指令

对于ydExtendedApi的用户而言，大多数情况下可以调用autoCreateCombPosition来自动完成组合。目前，自动指令只支持大商所和广期所，而上交所和深交所昨平仓时必须要先解组，而解组必须使用原生指令，因此，对于上交所和深交所的组合和解组，请直接使用原生指令。

```
virtual const YDCombPositionDef *autoCreateCombPosition(const int *combTypes)
```

每次调用autoCreateCombPosition，会首先按照combTypes指定的组合类型顺序，同一个组合类型中按照YDCombPositionDef.Priority（越小表示优先级越高）指定的优先级顺序，逐一检查可以组合的持仓，一旦检查到可以组合的持仓，就发送组合指令到柜台。请注意，每次调用autoCreateCombPosition最多只会组合一套，在生产实践中，可以用一个单独的线程按照一定的间隔持续调用本函数，如果找到需要组合的持仓并发送成功的，函数返回发送组合的传统组合持仓定义，如果没有找到需要组合的持仓，函数返回NULL。

autoCreateCombPosition的调用方法如下所示。

参数	说明
const int* combTypes	需要参与组合的类型数组，以-1结尾。组合时按照数组顺序逐个类型组合。 当为NULL时，默认为按照大商所所有传统组合持仓类型的顺序，即从YD_CPT_DCE_FuturesOffset到YD_CPT_DCE_SellOptionsCovered

组合成功或者失败后，会通过以下回调接口通知投资者。

```
virtual void notifyCombPositionOrder(const YDOrder *pOrder,const YDCombPositionDef
*pCombPositionDef,const YDAccount *pAccount) {}
```

7.5.3 自动工具

早期易达提供了独立的自动组合工具ydAutoCP供投资者使用，该工具通过命令行启动，启动后会按照一定的时间间隔搜索可以组合的持仓并发送组合指令。鉴于ydAutoCP有诸多限制条件和使用上的不便，更推荐投资者使用API来发送组合指令。出于与自动指令类似相同的原因，ydAutoCP也只支持大商所和广期所的自动组合。

ydAutoCP的配置文件样ydACP.ini的例如下：

```
AppID=yd_dev_1.0
AuthCode=ecbf8f06469eba63956b79705d95a603

AccountID=001
Password=001

# time ranges for auto combine position , can be multiple, no such setting indicates selecting at any time. Default is no setting
TimeRange=21:00:05-23:29:55
TimeRange=09:00:05-10:14:55
TimeRange=10:30:05-11:29:55
TimeRange=13:30:05-15:29:55

# refresh period, in seconds, default is 10, must be greater than 1
RefreshPeriod=10

# directory to hold log files, default is "ACPLog"
LogDir=ACPLog

# Comb positions types to be processed. If not given, default is all types. Optional values are as follow:
# YD_CPT_DCE_FuturesOffset=0
# YD_CPT_DCE_OptionsOffset=1
# YD_CPT_DCE_FuturesCalendarSpread=2
# YD_CPT_DCE_FuturesProductSpread=3
# YD_CPT_DCE_BuyOptionsVerticalSpread=4
# YD_CPT_DCE_SellOptionsVerticalSpread=5
# YD_CPT_DCE_OptionsStraddle=7
# YD_CPT_DCE_OptionsStrangle=8
# YD_CPT_DCE_BuyOptionsCovered=9
# YD_CPT_DCE_SellOptionsCovered=10
# YD_CPT_GFEX_FuturesOffset=0
# YD_CPT_GFEX_OptionsOffset=1
# YD_CPT_GFEX_FuturesCalendarSpread=2
# YD_CPT_GFEX_FuturesProductSpread=3
# YD_CPT_GFEX_BuyOptionsVerticalSpread=4
# YD_CPT_GFEX_SellOptionsVerticalSpread=5
# YD_CPT_GFEX_OptionsStraddle=7
# YD_CPT_GFEX_OptionsStrangle=8
# YD_CPT_GFEX_BuyOptionsCovered=9
# YD_CPT_GFEX_SellOptionsCovered=10
CombPositionType=0,1,2,3,4,5,7,8,9,10
```

除了几个显而易见的参数外，核心参数的说明如下：

- TimeRange指定了系统运行的时间。因为ydAutoCP遇到组合错误时会禁止该传统组合持仓定义参与后续组合，如果在非交易时间发送组合单会导致在此期间的传统组合持仓定义全部被禁止，因此，在配置系统时必须避开交易所的非交易时间段。同时，运行ydAutoCP的操作系统必须正确对时。
- CombPositionType与autoCreateCombPosition的参数combTypes的含义是完全相同的，请参考自动指令中的相关内容。

ydAutoCP的启动方式为，其中config.txt为易达API的配置文件，而ydAutoCP的配置文件ydACP.ini是默认加载的，因此该文件名不可以被修改：

```
./ydAutoCP config.txt
```

一旦发现系统发生了传统组合持仓定义被禁止的情况，如果想要使得该定义重新生效，必须重启ydAutoCP。传统组合持仓定义被禁止可以通过查看日志检查，相关错误信息样例为：

```
error in handling (pg2302&-eg2303,1) for account 12345678
```

7.6 行权与履约

下文将详细介绍各个交易所的行权与履约业务，虽然本文尽可能全面阐述具体业务逻辑，但是难免有所疏漏，且随着交易所的业务发展，以下业务逻辑可能将不再符合交易所行权与履约业务的实际。因此，下文内容不可以作为投资者实施交易所行权与履约业务的依据，一切以交易所公布为准。

为避免行权与履约过程中造成损失，在生产环境中使用易达行权与履约API接口开展业务前，**务必**在交易所仿真环境中测试与验证行权履约行为是否符合预期，易达不对因为未正确理解、未测试的行为造成的行权履约损失承担任何责任。

7.6.1 上期所和能源所行权与履约

根据上期所和能源所行权履约业务规则，行权前投资者可以通过平仓或者双向期权持仓对冲两种交易手段了结期权持仓；到期日当天交易所会按照实值期权默认行权、虚值期权默认不行权的逻辑帮助期权买方自动行权，但实值期权买方可以放弃行权、虚值期权买方可以申请行权以覆盖交易所的自动行权逻辑。同时，到期日当天期权买方和卖方可以申请行权履约后对冲双向期货持仓。若投资者的某持仓同时使用了上述业务，那么交易所的处理逻辑为：首先处理双向期权持仓对冲，其次处理期权行权申请的期货建仓和放弃行权，然后对剩余持仓进行自动行权或自动放弃处理，再次处理行权履约后双向期货持仓对冲。上述业务均属申请类业务，即盘中只做资金和持仓冻结处理，由盘后结算系统统一计算处理。

- 双向期权持仓对冲是对同一交易编码下的同合约双向期权持仓申请对冲平仓或放弃自动对冲平仓。对于普通投资者，默认不对冲双向期权持仓，需要投资者手工申请；对于做市商，默认对冲双向期权持仓，若做市商不希望自动对冲双向期权持仓，可以申请放弃自动对冲。双向期权持仓对冲结果从当日期权持仓量中扣除，收取交易手续费，持仓量相应调整。欧式期权的对冲申请和放弃对冲申请时间为到期日15:30前，美式期权的对冲申请和放弃对冲申请时间为到期日前任一交易日交易时间和到期日15:30前。申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效。
- 行权和放弃行权是指欧式期权买方可以在到期日15:30 前提交行权申请、放弃行权申请；美式期权买方可以在到期日前任一交易日交易时段提交行权申请，到期日当天可以在15:30之前提交行权申请、放弃行权申请。申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效。
- 到期日自动行权是指到期日结算前，对投资者未在规定时间内提交行权或放弃行权申请、且相对标的期货合约结算价有实值额的期权买持仓做自动行权处理，其他期权持仓做自动放弃处理。
- 行权（履约）后期货持仓自对冲是指期权买方（卖方）可以申请对其同一交易编码下行权（履约）获得的双向期货持仓进行对冲平仓，也可以申请对其同一交易编码下行权（履约）获得的期货持仓与期货市场原有持仓进行对冲平仓，对冲数量不超过行权（履约）获得的期货持仓量。对冲结果从当日期货持仓量中扣除。欧式期权的申请时间为到期日 15:30 前；美式期权的申请时间为到期日前任一交易日交易时间和到期日 15:30 前。申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效。交易所接口中，行权后双向期货持仓对冲标志是在行权指令中指定的，为了简化易达API结构，易达柜台固定设置了行权后双向期货持仓对冲标志，投资者不可修改，因此只要期权买方向交易所发送行权申请，即表示需要行权后双向期货持仓对冲。履约后双向期货持仓对冲标志通过专有指令发送。

7.6.1.1 上期所和能源所双向期权持仓对冲

使用insertOrder指令向交易所发起双向期权持仓对冲或者放弃双向期权持仓对冲的申请，请求参数如下表所示。对于同一投资者、合约、对冲类型（OrderType）和投保标志的对冲申请，只保留最近申请的记录。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_OptionSelfClose

参数	字段	说明
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	OrderType	YD_ODT_CloseSelfOptionPosition：申请双向期权持仓对冲，供普通投资者使用 YD_ODT_ReserveOptionPosition：放弃双向期权持仓对冲，供做市商使用
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
	OrderVolume	发起并指定请求的数量
YDInstrument		指定待双向期权持仓对冲的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

期权对冲申请成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作申请报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	固定为YD_YOF_OptionSelfClose
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	OrderSysID	交易所返回的系统报单号，若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	交易所返回的系统报单号，不会截取交易所报单编号，保留全精度
	InsertTime	从该交易日开始（17点整）到报单时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	InsertTimeStamp	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	OrderStatus	申请成功为YD_OS_AllTraded，申请失败则为YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待双向期权持仓对冲的期权合约指针
YDAccount		当前API登录账户

由于上期所和能源所对于同一投资者、合约、对冲类型（OrderType）和投保标志的对冲申请，只保留最近申请的记录，因此，投资者通过指定和双向期权持仓对冲申请相同的合约、对冲类型和投保标志的报单以撤销双向期权持仓对冲申请。为了符合交易所指令的特性，易达撤销双向期权持仓对冲使用与期权双向期权持仓对冲申请期权对冲独立的报单，因此，撤销指令中的OrderRef不应填写申请指令中的OrderRef，以免受到报单号单调递增检查和报单组限制从而导致撤销指令发送失败。

使用insertOrder指令向交易所撤销双向期权持仓对冲申请，请求参数如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_OptionSelfClose
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	OrderType	填写待撤销对冲申请对应的OrderType
	HedgeFlag	填写待撤销对冲申请对应的HedgeFlag
	OrderVolume	填0，表示撤销期权对冲申请
YDInstrument		指定待撤销待双向期权持仓对冲的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

撤销双向期权持仓对冲成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	固定为YD_YOF_OptionSelfClose
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	OrderSysID	交易所返回的系统报单号，若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	交易所返回的系统报单号，不会截取交易所报单编号，保留全精度
	InsertTime	从该交易日开始（17点整）到报单时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	InsertTimeStamp	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	OrderStatus	撤销成功为YD_OS_AllTraded，撤销失败则为YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待撤销待双向期权持仓对冲的期权合约指针

参数	字段	说明
YDAccount		当前API登录账户

7.6.1.2 上期所和能源所行权

为便于编写程序，易达为是否支持期权行权提供了参数化表示，使用YDExchange.OptionExecutionSupport来表示对期权行权不同程度的支持：

- 0：不支持。目前中金所属于这个分类。
- 1：支持但不包含风控，所谓不风控就是不做钱仓检查和冻结。目前没有交易所属于这个分类。
- 2：支持且包含风控，所谓风控就是做钱仓检查和冻结。目前上期所、能源所、大商所、郑商所、广期所、上交所、深交所属于这个分类。

使用insertOrder指令向交易所发起行权申请。参数填写方法如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_OptionExecute
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	Direction	填写YD_D_Sell
	OffsetFlag	填写行权对应仓位的OffsetFlag。 对于上期所和能源所，今仓填YD_OF_CloseToday，昨仓填YD_OF_CloseYesterday 对于大商所、郑商所、广期所，填写YD_OF_Close
	OrderType	填写YD_ODT_Limit
	HedgeFlag	投保标志。 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderVolume	指定行权数量
YDInstrument		指定待行权的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

期权行权回报通过notifyOrder返回。若行权申请成功，回报会显示为挂单；若行权申请失败，回报则显示为错单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	固定为YD_YOF_OptionExecute
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	OrderSysID	交易所返回的系统报单号，若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	交易所返回的系统报单号，不会截取交易所报单编号，保留全精度

参数	字段	说明
	InsertTime	从该交易日开始（17点整）到报单时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	InsertTimeStamp	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	OrderStatus	申请成功为YD_OS_Queuing，申请失败则为YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待行权的期权合约指针
YDAccount		当前API登录账户

由于期权行权申请成功后，申请委托处于挂单状态，因此，如果要撤销行权申请，可以使用cancelOrder撤销。撤销行权申请和撤销普通报单的用法以及回报是相同的，具体请参考撤单。

7.6.1.3 上期所和能源所放弃行权

为便于编写程序，易达为是否支持使用YD_YOF_OptionAbandonExecute放弃期权行权提供了参数化表示，使用YDExchange.OptionAbandonExecutionSupport来表示对期权行权不同程度的支持：

- 0：不支持。目前中金所、上交所、深交所、大商所、广期所属于这个分类。中金所、上交所、深交所不支持放弃期权行权指令，大商所、广期所不支持使用YD_YOF_OptionAbandonExecute放弃期权行权，但是可以使用YD_YOF_Mark放弃期权行权，具体方法请参考下文。
- 1：支持但不包含风控，所谓不风控就是不做钱仓检查和冻结。目前没有交易所属于这个分类。
- 2：支持且包含风控，所谓风控就是做钱仓检查和冻结。目前上期所、能源所、郑商所属于这个分类。

使用insertOrder指令向交易所发起放弃行权申请。参数填写方法如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_OptionAbandonExecute
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	Direction	填写YD_D_Sell
	OffsetFlag	填写行权对应仓位的OffsetFlag。 对于上期所和能源所，今仓填YD_OF_CloseToday，昨仓填YD_OF_CloseYesterday 对于其他交易所，填写YD_OF_Close
	OrderType	填写YD_ODT_Limit

参数	字段	说明
	HedgeFlag	投保标志，期货交易所与股票期权交易所的定义不同。 期货交易所如下： YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 股票期权交易所如下： YD_HF_Normal=1：普通 YD_HF_Covered=3：备兑
	OrderVolume	指定行权数量
YDInstrument		指定待放弃行权的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

期权放弃行权回报通过notifyOrder返回。若放弃行权申请成功，回报会显示为挂单；若放弃行权申请失败，回报则显示为错单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	固定为YD_YOF_OptionAbandonExecute
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	OrderSysID	交易所返回的系统报单号，若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	交易所返回的系统报单号，不会截取交易所报单编号，保留全精度
	InsertTime	从该交易日开始（17点整）到报单时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	InsertTimeStamp	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	OrderStatus	申请成功为YD_OS_Queuing，申请失败则为YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待放弃行权的期权合约指针
YDAccount		当前API登录账户

由于期权放弃行权申请成功后，申请委托处于挂单状态，因此，如果要撤销放弃行权申请，可以使用cancelOrder撤销。撤销放弃行权申请和撤销普通报单的用法以及回报是相同的，具体请参考撤单。

7.6.1.4 上期所和能源所履约后双向期货持仓对冲

使用insertOrder指令向交易所发起期履约后双向期货持仓对冲请求，请求参数如下表所示。对于同一投资者、合约、对冲类型（OrderType）和投保标志的对冲申请，只保留最近申请的记录。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_OptionSelfClose
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	OrderType	YD_ODT_SellCloseSelfFuturesPosition：请求履约后双向期货持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
	OrderVolume	双向期货持仓对冲数量
YDInstrument		指定待履约后双向期货持仓对冲的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

履约后双向期货持仓对冲回报、撤销履约后双向期货持仓对冲申请以及撤销履约后双向期货持仓对冲申请回报与上期所和能源所双向期权持仓对冲类似，请参考对应内容。

7.6.2 大商所行权与履约

> **Important**
>
> 2024年12月31日结算后，大商所升级交易系统，升级后“API暂无法通过交易系统提交客户的双向期权持仓对冲平仓申请、行权后双向期货持仓对冲平仓申请、履约后双向期货持仓对冲平仓申请。上述业务请使用会服办理。后续交易所将组织市场升级，完成后上述功能即会恢复，相关工作安排另行通知。在此期间请各会员及时提醒客户相关操作变动，并做好相关服务。”。
>
> 大商所完成升级后，易达柜台的大商所双向期权持仓对冲、大商所履约后双向期货持仓对冲功能将无法使用，若申报则会收到交易所错单回报。
>
> 自1.486.96.62起，易达柜台取消了在行权指令中默认设置的行权后双向期货持仓对冲标志，投资者可暂时通过会服提交行权后双向期货持仓对冲平仓申请。

根据大商所行权履约业务规则，行权前投资者可以通过平仓或者双向期权持仓对冲两种交易手段了结期权持仓；到期日当天交易所会按照实值期权默认行权、虚值期权默认不行权的逻辑帮助期权买方自动行权，但实值期权买方可以放弃行权、虚值期权买方可以申请行权以覆盖交易所的自动行权逻辑。同时，到期日当天期权买方可以申请行权后对冲双向期货持仓。在任意交易日，投资者可以申请履约后对冲双向期货持仓，一旦申请成功永久有效。若投资者的某持仓同时使用了上述业务，那么交易所的处理逻辑为：首先处理双向期权持仓对冲，其次处理期权行权期货建仓，再次处理行权后双向期货持仓对冲，最后处理履约后双向期货持仓对冲。双向期权持仓对冲和放弃行权业务均属特殊申请类业务，盘中不做资金和持仓冻结处理；行权申请业务属一般申请类业务，盘中冻结资金和持仓冻结处理。

- 双向期权持仓对冲是申请对同一交易编码下的同合约双向期权持仓进行对冲平仓，遵循最大对冲数量（买持仓量与卖持仓量取小）且先平投机持仓后平套保持仓的原则。对冲平仓的平仓价格为当日结算价，平仓顺序为先开先平，计入成交量和成交额，收取交易手续费，持仓量相应调整。申请时间为交易日的交易时间及到期日的15:00-15:30。申请对象为指定交易编码下的指定合约，该申请仅当日有效。
- 到期日自动行权是指到期日闭市后，行权价格小于（大于）标的期货合约当日结算价的看涨（跌）期权持仓自动申请行权，其他期权自动放弃行权。对于交易所自动申请行权的持仓，投资者可以放弃自动申请行权；如果不放弃自动行权，交易所将替买方按照全部持仓手数（不扣除买方已提交的行权申请手数）下达行权申请；如果既提交了行权申请，又未放弃自动行权，交易所将先处理买方提交的行权申请，后处理交易所提交的自动行权申请，直至买方持仓全部行权；如

果放弃自动行权，交易所将不会替买方对该合约下达行权申请；如需对该合约的部分持仓行权，需提交相应手数的行权申请并取消自动申请行权（先后顺序无影响）。对于交易所自动放弃行权的持仓，投资者可以申请权行权覆盖自动放弃行权。申请时间为到期日的交易时间及15:00-15:30。手工申请行权的申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效；手工放弃行权的申请对象为指定交易编码下的指定合约，该申请仅当日有效。请注意，手工申请行权和手工放弃行权可以同时存在，并不冲突。下面假设某投资者有10手持仓，在不同条件的排列组合下，最终行权结果如下表所示，为求简单，以下行权结果不考虑因为资金不足等无法行权的情况。

交易所自动逻辑	手工申请行权	手工放弃行权	行权结果
行权10手	申请5手	是	行权5手
行权10手	申请5手	否	行权10手
行权10手	不申请	是	不行权
行权10手	不申请	否	行权10手
放弃行权	申请5手	是	行权5手
放弃行权	申请5手	否	行权5手
放弃行权	不申请	是	不行权
放弃行权	不申请	否	不行权

- 行权后双向期货持仓对冲是指期权买方可以申请对其同一交易编码下行权后的双向期货持仓进行对冲平仓，遵循对冲数量不超过行权获得的期货持仓量（行权建仓量与反向持仓量取小）且先平投机持仓后平套保持仓的原则。对冲平仓的平仓价格为当日结算价，平仓顺序为先开先平，收取交易手续费，持仓量相应调整。申请时间为交易日的交易时间及到期日的15:00-15:30。申请对象为指定交易编码下的指定合约，该申请仅当日有效。交易所接口中，行权后双向期货持仓对冲标志是在行权指令中指定的，为了简化易达API结构，易达柜台固定设置了行权后双向期货持仓对冲标志，投资者不可修改。
- 履约后双向期货持仓对冲是指期权卖方可以申请对其同一交易编码下履约后的双向期货持仓进行对冲平仓，履约后双向期货持仓对冲逻辑同行权后双向期货持仓对冲逻辑。申请时间为交易日的交易时间及到期日的15:00-15:30。申请对象为指定交易编码，申请成功后永久有效。

7.6.2.1 大商所双向期权持仓对冲

使用insertOrder指令向交易所发起双向期权持仓对冲申请，请求参数如下表所示。需注意，大商所的双向期权持仓对冲指令无法选择席位报单，易达交易系统会一律覆盖指定为从首席位发出。原因是这类报单既没有系统报单号，也没有本地报单号，所以无法实现任何识别重复接收的机制。如果使用全管理席位叠加全收模式流水，所有席位都会收到回报，会造成易达柜台产生无法关联的报单回报，所以，目前的实现是一律用首席位发送和接收。但是这也带来一个问题，如果首席位不是公用席位，那么对于不能使用首席位的投资者，就无法发出本指令，因此对于大商所，不能将首席位配置为专属席位。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Buy：发起双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲

参数	字段	说明
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定的待双向期权持仓对冲合约指针
YDAccount		填写NULL，表示使用当前API登录账户

由于大商所双向期权持仓对冲指令的特殊性，易达无法关联发往交易所的请求和回报，易达柜台在收到交易所的回报后，会自行生成特殊回报，如下表所示。因此，投资者在发送大商所的双向期权持仓对冲指令后，不要在客户端保留报单或者等待与之匹配的回报，而应在收到YDOrderFlag为YD_YOF_Mark的报单回报后，根据合约或合约所在交易所、OrderType和OrderStatus来区分具体发生的业务和成功与否。也是基于上述同样的原因，即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，大商所的对冲指令也不会发送柜台回报。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Buy：发起双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderRef	-1
	OrderLocalID	-1
	OrderSysID	0
	InsertTime	-1
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待双向期权持仓对冲合约指针
YDAccount		当前API登录账户

使用insertOrder指令向交易所撤销双向期权持仓对冲申请，请求参数如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Sell：撤销双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲

参数	字段	说明
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定的待双向期权持仓对冲合约指针
YDAccount		填写NULL，表示使用当前API登录账户

撤销双向期权持仓对冲成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Sell：撤销双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderRef	-1
	OrderLocalID	-1
	OrderSysID	0
	InsertTime	-1
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待双向期权持仓对冲合约指针
YDAccount		当前API登录账户

7.6.2.2 大商所行权

大商所行权的报送方式与上期所和能源所行权相同，详情请参考上期所和能源所行权。

7.6.2.3 取消大商所到期日自动行权

请参考以下说明使用insertOrder向交易所发起取消到期日自动行权申请。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark

参数	字段	说明
	OrderRef	客户报单编号 由于大商所放弃自动行权的上下行接口均无报单号相关字段，导致易达无法知晓放弃行权的交易所回报对应的投资者报单，因此柜台发送回报的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Buy：放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定待放弃自动行权的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

由于大商所放弃行权指令的特殊性，易达无法关联发往交易所的请求和回报，易达柜台在收到交易所的回报后，会自行生成特殊回报，如下表所示。因此，投资者在发送大商所的放弃行权指令后，不要在客户端保留报单或者等待与之匹配的回报，而应在收到YDOrderFlag为YD_YOF_Mark的报单回报后，根据合约或合约所在交易所、OrderType和OrderStatus来区分具体发生的业务和成功与否。也是基于上述同样的原因，即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，大商所的放弃行权指令也不会发送柜台回报。

放弃行权成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Buy：放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号或者柜台错误号
YDInstrument		待放弃自动行权的期权合约指针
YDAccount		当前API登录账户

使用insertOrder指令向交易所撤销放弃行权申请，请求参数如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark

参数	字段	说明
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Sell：撤销放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定的待撤销放弃自动行权的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

由于大商所撤销放弃行权指令的特殊性，易达无法关联发往交易所的请求和回报，易达柜台在收到交易所的回报后，会自行生成特殊回报，如下表所示。因此，投资者在发送大商所的撤销放弃行权指令后，不要在客户端保留报单或者等待与之匹配的回报，而应在收到YDOrderFlag为YD_YOF_Mark的报单回报后，根据合约或合约所在交易所、OrderType和OrderStatus来区分具体发生的业务和成功与否。也是基于上述同样的原因，即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，大商所的放弃行权指令也不会发送柜台回报。

撤销放弃行权成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Sell：撤销放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待撤销放弃自动行权的期权合约指针
YDAccount		当前API登录账户

7.6.2.4 大商所履约后双向期货持仓对冲

出于简化API考虑，在发起履约对冲指令时，请任意指定一个属于大商所或者广期所的合约，便于易达柜台通过该合约找到要发起履约对冲指令的交易所。

请参考以下方法使用insertOrder指令向交易所发起履约后双向期货持仓对冲申请。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Buy：申请对冲请求
	OrderType	YD_ODT_CloseFuturesPositionMark：履约后双向期货持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定对应交易所的任意合约指针
YDAccount		填写NULL，表示使用当前API登录账户

在履约后双向期货持仓对冲回报中，交易所在返回时并不包含具体合约，因此，易达会将对应交易所的首个合约随回报传回，投资者只需要获取该合约的交易所即可。履约后双向期货持仓对冲回报、撤销履约后双向期货持仓对冲和撤销履约后双向期货持仓对冲回报的其他内容与大商所双向期权持仓对冲类似，请参考对应内容。

7.6.3 广期所行权与履约

根据广期所行权履约业务规则，行权前投资者可以通过平仓或者双向期权持仓对冲两种交易手段了结期权持仓；到期日当天交易所会按照实值期权默认行权、虚值期权默认不行权的逻辑帮助期权买方自动行权，但实值期权买方可以放弃行权、虚值期权买方可以申请行权以覆盖交易所的自动行权逻辑。同时，到期日当天期权买方可以申请行权后对冲双向期货持仓。在任意交易日，投资者可以申请履约后对冲双向期货持仓，一旦申请成功永久有效。若投资者的某持仓同时使用了上述业务，那么交易所的处理逻辑为：首先处理双向期权持仓对冲，其次处理期权行权期货建仓，再次处理行权后双向期货持仓对冲，最后处理履约后双向期货持仓对冲。双向期权持仓对冲和放弃行权业务均属特殊申请类业务，盘中不做资金和持仓冻结处理；行权申请业务属一般申请类业务，盘中冻结资金和持仓冻结处理。

- 双向期权持仓对冲是申请对同一交易编码下的同合约双向期权持仓进行对冲平仓，遵循最大对冲数量（买持仓量与卖持仓量取小）且先平投机持仓后平套保持仓的原则。对冲平仓的平仓价格为当日结算价，平仓顺序为先开先平，计入成交量和成交额，收取交易手续费，持仓量相应调整。申请时间为交易日的交易时间及到期日的15:00-15:30。申请对象为指定交易编码下的指定合约，该申请仅当日有效。
- 到期日自动行权是指到期日闭市后，行权价格小于（大于）标的期货合约当日结算价的看涨（跌）期权持仓自动申请行权，其他期权自动放弃行权。对于交易所自动申请行权的持仓，投资者可以放弃自动申请行权；如果不放弃自动行权，交易所将替买方按照全部持仓手数（不扣除买方已提交的行权申请手数）下达行权申请；如果既提交了行权申请，又未放弃自动行权，交易所将先处理买方提交的行权申请，后处理交易所提交的自动行权申请，直至买方持仓全部行权；如果放弃自动行权，交易所将不会替买方对该合约下达行权申请；如需对该合约的部分持仓行权，需提交相应手数的行权申请并取消自动申请行权（先后顺序无影响）。对于交易所自动放弃行权的持仓，投资者可以申请权行权覆盖自动放弃行权。申请时间为到期日的交易时间及15:00-15:30。手工申请行权的申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效；手工放弃行权的申请对象为指定交易编码下的指定合约，该申请仅当日有效。请注意，手工申请行权和手工放弃行权可以同时存在，并不冲突。下面假设某投资者有10手持仓，在不同条件的排列组合下，最终行权结果如下表所示，为求简单，以下行权结果不考虑因为资金不足等无法行权的情况。

交易所自动逻辑	手工申请行权	手工放弃行权	行权结果
行权10手	申请5手	是	行权5手
行权10手	申请5手	否	行权10手
行权10手	不申请	是	不行权

交易所自动逻辑	手工申请行权	手工放弃行权	行权结果
行权10手	不申请	否	行权10手
放弃行权	申请5手	是	行权5手
放弃行权	申请5手	否	行权5手
放弃行权	不申请	是	不行权
放弃行权	不申请	否	不行权

- 行权后双向期货持仓对冲是指期权买方可以申请对其同一交易编码下行权后的双向期货持仓进行对冲平仓，遵循对冲数量不超过行权获得的期货持仓量（行权建仓量与反向持仓量取小）且先平投机持仓后平套保持仓的原则。对冲平仓的平仓价格为当日结算价，平仓顺序为先开先平，收取交易手续费，持仓量相应调整。申请时间为交易日的交易时间及到期日的15:00-15:30。申请对象为指定交易编码下的指定合约，该申请仅当日有效。交易所接口中，行权后双向期货持仓对冲标志是在行权指令中指定的，为了简化易达API结构，易达柜台固定设置了行权后双向期货持仓对冲标志，投资者不可修改。
- 履约后双向期货持仓对冲是指期权卖方可以申请对其同一交易编码下履约后的双向期货持仓进行对冲平仓，履约后双向期货持仓对冲逻辑同行权后双向期货持仓对冲逻辑。申请时间为交易日的交易时间及到期日的15:00-15:30。申请对象为指定交易编码，申请成功后永久有效。

7.6.3.1 广期所双向期权持仓对冲

使用insertOrder指令向交易所发起双向期权持仓对冲申请，请求参数如下表所示。需注意，广期所的双向期权持仓对冲指令无法选择席位报单，易达交易系统会一律覆盖指定为从首席位发出。原因是这类报单既没有系统报单号，也没有本地报单号，所以无法实现任何识别重复接收的机制。如果使用全管理席位叠加全收模式流水，所有席位都会收到回报，会造成易达柜台产生无法关联的报单回报，所以，目前的实现是一律用首席位发送和接收。但是这也带来一个问题，如果首席位不是公用席位，那么对于不能使用首席位的投资者，就无法发出本指令，因此对于广期所，不能将首席位配置为专属席位。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Buy：发起双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定的待双向期权持仓对冲合约指针
YDAccount		填写NULL，表示使用当前API登录账户

由于广期所双向期权持仓对冲指令的特殊性，易达无法关联发往交易所的请求和回报，易达柜台在收到交易所的回报后，会自行生成特殊回报，如下表所示。因此，投资者在发送广期所的双向期权持仓对冲指令后，不要在客户端保留报单或者等待与之匹配的回报，而应在收到YDOrderFlag为YD_YOF_Mark的报单回报后，根据合约或合约所在交易所、OrderType和OrderStatus来区分具体发生的业务和成功与否。也是基于上述同样的原因，即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，广期所的对冲指令也不会发送柜台回报。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Buy：发起双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderRef	-1
	OrderLocalID	-1
	OrderSysID	0
	InsertTime	-1
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待双向期权持仓对冲合约指针
YDAccount		当前API登录账户

使用insertOrder指令向交易所撤销双向期权持仓对冲申请，请求参数如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Sell：撤销双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定的待双向期权持仓对冲合约指针
YDAccount		填写NULL，表示使用当前API登录账户

撤销双向期权持仓对冲成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark

参数	字段	说明
	Direction	YD_D_Sell：撤销双向期权持仓对冲申请
	OrderType	YD_ODT_PositionOffsetMark：双向期权持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderRef	-1
	OrderLocalID	-1
	OrderSysID	0
	InsertTime	-1
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待双向期权持仓对冲合约指针
YDAccount		当前API登录账户

7.6.3.2 广期所行权

广期所行权的报送方式与上期所和能源所行权相同，详情请参考上期所和能源所行权。

7.6.3.3 取消广期所到期日自动行权

请参考以下说明使用insertOrder向交易所发起取消到期日自动行权申请。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	客户报单编号 由于广期所放弃自动行权的上下行接口均无报单号相关字段，导致易达无法知晓放弃行权的交易所回报对应的投资者报单，因此柜台发送回报的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Buy：放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定待放弃自动行权的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

由于广期所放弃行权指令的特殊性，易达无法关联发往交易所的请求和回报，易达柜台在收到交易所的回报后，会自行生成特殊回报，如下表所示。因此，投资者在发送广期所的放弃行权指令后，不要在客户端保留报单或者等待与之匹配的回报，而应在收到YDOrderFlag为YD_YOF_Mark的报单回报后，根据合约或合约所在交易所、OrderType和OrderStatus来区分具体发生的业务和成功与否。也是基于上述同样的原因，即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，广期所的放弃行权指令也不会发送柜台回报。

放弃行权成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Buy：放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号或者柜台错误号
YDInstrument		待放弃自动行权的期权合约指针
YDAccount		当前API登录账户

使用insertOrder指令向交易所撤销放弃行权申请，请求参数如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Sell：撤销放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定的待撤销放弃自动行权的期权合约指针
YDAccount		填写NULL，表示使用当前API登录账户

由于广期所撤销放弃行权指令的特殊性，易达无法关联发往交易所的请求和回报，易达柜台在收到交易所的回报后，会自行生成特殊回报，如下表所示。因此，投资者在发送广期所的撤销放弃行权指令后，不要在客户端保留报单或者等待与之匹配的回报，而应在收到YDOrderFlag为YD_YOF_Mark的报单回报后，根据合约或合约所在交易所、OrderType和OrderStatus来区分具体发生的业务和成功与否。也是基于上述同样的原因，即便YDAccount中的AccountFlag设置了柜台回报YD_AF_NotifyOrderAccept，广期所的放弃行权指令也不会发送柜台回报。

撤销放弃行权成功或失败的回报会通过notifyOrder返回，无论成功或者失败，对应报单均进入终态，无法继续操作撤销报单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	YD_YOF_Mark
	Direction	YD_D_Sell：撤销放弃行权
	OrderType	YD_ODT_OptionAbandonExecuteMark：放弃行权
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
	OrderStatus	成功：YD_OS_AllTraded 失败：YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument		待撤销放弃自动行权的期权合约指针
YDAccount		当前API登录账户

7.6.3.4 广期所履约后双向期货持仓对冲

出于简化API考虑，在发起履约对冲指令时，请任意指定一个属于大商所或者广期所的合约，便于易达柜台通过该合约找到要发起履约对冲指令的交易所。

请参考以下方法使用insertOrder指令向交易所发起履约后双向期货持仓对冲申请。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Mark
	OrderRef	始终填写0 回报中的OrderRef始终为-1，和请求中的OrderRef无法对应
	Direction	YD_D_Buy：申请对冲请求
	OrderType	YD_ODT_CloseFuturesPositionMark：履约后双向期货持仓对冲
	HedgeFlag	投保标志 YD_HF_Speculation=1：投机 YD_HF_Hedge=3：套保 交易所不使用投保标志，易达系统使用资金账户和投保标志唯一确定交易编码，因此请指定对应的投标标志
YDInstrument		指定对应交易所的任意合约指针
YDAccount		填写NULL，表示使用当前API登录账户

在履约后双向期货持仓对冲回报中，交易所在返回时并不包含具体合约，因此，易达会将对应交易所的首个合约随回报传回，投资者只需要获取该合约的交易所即可。履约后双向期货持仓对冲回报、撤销履约后双向期货持仓对冲和撤销履约后双向期货持仓对冲回报的其他内容与广期所双向期权持仓对冲类似，请参考对应内容。

7.6.4 郑商所行权与履约

根据郑商所行权履约业务规则，行权前投资者可以通过平仓了结期权持仓；到期日当天交易所会按照实值期权默认行权、虚值期权默认不行权的逻辑帮助期权买方自动行权，但实值期权买方可以放弃行权、虚值期权买方可以申请行权以覆盖交易所的自动行权逻辑。交易所的处理逻辑为：首先处理期权行权申请的期货建仓和放弃行权，然后对剩余持仓进行自动行权或自动放弃处理。上述业务均属申请类业务，即盘中只做资金和持仓冻结处理，由盘后结算系统统一计算处理。

- 行权和放弃行权是指期权买方可以在到期日及到期日前任意交易时间段提交行权申请、放弃行权申请。申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效。
- 到期日自动行权是到期日结算时，对未在规定时间内提交行权或放弃申请的期权持仓，按实值期权（期货结算价）自动行权，虚值期权自动放弃行权。具体处理如下：行权价格小于当日标的物结算价的看涨期权（买权）持仓自动行权；行权价格大于当日标的物结算价的看跌期权（卖权）持仓自动行权；其他期权持仓自动放弃。需要注意的是，浅实值和浅虚值期权由于标的期货收盘价与 结算价的不同、手续费差异和投资者对市场未来走势的判断，因此，可能需要对浅实值期权提交放弃或对浅虚值期权提交行权。

7.6.4.1 郑商所行权

郑商所行权的报送方式与上期所和能源所行权相同，详情请参考上期所和能源所行权。

7.6.4.2 郑商所放弃行权

郑商所放弃行权的报送方式与上期所和能源所行权相同，详情请参考上期所和能源所放弃行权。

7.6.5 中金所行权与履约

除国债期权外的中金所期权不支持行权指令，在最后交易日中金所会自动行权所有实值期权，如果不想参与行权，请于最后交易日前平仓。

对于国债期权，行权前投资者可以通过平仓或者双向期权持仓对冲两种交易手段了结期权持仓；期权买方可在到期日前申请行权，到期日当天申请行权或者放弃行权，对于没有提交任何行权或者放弃行权申请的持仓，交易所按照规则自动行权或者放弃行权。若投资者的某持仓同时使用了上述业务，那么交易所的处理逻辑为：首先处理双向期权持仓对冲，其次处理期权行权申请的期货建仓和放弃行权，然后对剩余持仓进行自动行权或自动放弃处理，再次处理行权履约后双向期货持仓对冲。上述业务均属申请类业务，即盘中只做资金和持仓冻结处理，由盘后结算系统统一计算处理。

- 双向期权持仓对冲是结算时交易所对同一交易编码下期权合约双向持仓对冲平仓。申请需通过会员经参与人平台申请，一次申请持续有效。交易所默认不对冲。
- 行权和放弃行权是在到期日前的每个交易日，期权合约的买方可在交易时间向交易所提交行权申请；在到期日，期权合约的买方可在交易时间及交易结束后15分钟内向交易所提交行权或放弃申请。申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效。
- 到期日自动行权是指对于到期日未提交任何申请的未平仓期权合约，交易所对符合行权条件的买方持仓自动行权：买方提交行权最低盈利金额的，行权条件为实值额大于最低盈利金额；买方未提交行权最低盈利金额的，行权条件为实值额大于0；其他期权默认放弃行权。其中，$实值额 = 最后交易日期权合约结算价 \times 标的国债期货合约面值/100$。
- 行权（履约）后期货持仓自对冲是结算时交易所对同一交易编码下行权（履约）后期货双向持仓对冲平仓，对冲数量不超过行权（履约）获得的期货买持仓量和卖持仓量中的较大值。申请需通过会员经参与人平台申请，一次申请持续有效。交易所默认不对冲。

7.6.5.1 中金所行权

中金所国债期权行权的报送方式与上期所和能源所行权相同，详情请参考上期所和能源所行权。

其他产品不支持行权申请。

7.6.5.2 中金所放弃行权

中金所国债期权放弃行权的报送方式与上期所和能源所放弃行权相同，详情请参考上期所和能源所放弃行权。

其他产品不支持放弃行权申请。

7.6.6 上交所和深交所行权与履约

根据上交所和深交所行权履约业务规则，行权前投资者可以通过平仓了结期权持仓；行权日可申请行权和合并行权，即行权指令合并申报。若同时申请行权和合并行权，中国结算的处理逻辑为：先处理合并行权申报，后处理非合并行权申报。上述业务均属申请类业务，即盘中只做资金和持仓冻结处理，由盘后结算系统统一计算处理。

- 行权申请是指期权买方可以在行权日15:00-15:30时段提交行权申请。申请对象为指定交易编码下的指定合约和指定数量，该申请仅当日有效。投资者可行权数量应为未平仓权利仓数量（包括已申报未成交数量）。若未申请行权，则认为放弃行权。
- 合并行权（行权指令合并申报）是指可以同时对相同标的证券当日到期的认购和认沽期权权利仓合并申报行权，实现到期认购和认沽期权的同步行权，提高投资者的资金使用效率，避免交收期间现货隔夜风险。投资者可多次提交合并行权申请，累计申报的行权数量不应超过持有的权利仓净头寸（即相应衍生品合约账户同一合约到期组合策略解除后，权利仓和义务仓对冲后的持仓）。若某次申报数量超过当前净头寸，则该笔申报中仅持有的权利仓净头寸部分有效。合并行权申请时间为行权日9:15至9:25、9:30至11:30、13:00至15:30。合并行权的期权合约必须满足以下条件：
  - 期权合约标的证券应当相同，比如沪深300ETF期权合约与上证50ETF期权合约不能合并申报
  - 认沽期权行权价必须高于认购期权且必须是当日到期的期权合约
  - 合约单位必须相等。标的分红等会导致ETF期权的合约进行调整，会引起标准合约与非标准合约（合约代码非“M”的合约）的合约单位不同，而这类非标准合约与标准合约不能进行合并申报
  - 认购和认沽期权的行权数量不得超过投资者持有的相应合约权利仓净头寸。净头寸以当日到期组合策略解除后的净头寸为准。若某次申报数量超过当前可用的行权额度，该笔申报全部无效。

7.6.6.1 上交所与深交所行权

上交所与深交所行权的报送方式与上期所和能源所行权相同，详情请参考上期所和能源所行权。

7.6.6.2 上交所与深交所合并行权

对于上交所和深交所，请参考以下方法使用insertOptionExecTogetherOrder或者checkAndInsertOptionExecTogetherOrder指令向交易所发起合并行权申请，并使用cancelOrder撤销合并行权请求。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_OptionExecuteTogether
	OrderRef	客户报单编号，详情请参考普通报单中关于OrderRef的说明
	OrderVolume	指定合并行权的数量。
YDInstrument	pInstrument	指定待合并行权的期权合约指针
	pInstrument2	指定待合并行权的期权合约指针2
YDAccount	pAccount	填写NULL，表示使用当前API登录账户

合并行权回报通过notifyOptionExecTogetherOrder返回。若合并行权申请成功，回报会显示为挂单；若行权申请失败，回报则显示为错单。回报中的YDOrder与申请中的YDInputOrder相同字段的值保持一致，因此下表忽略与YDInputOrder相同字段以及无意义字段。投资者主要关注回报中的OrderStatus以及ErrorNo了解申请是否成功。

参数	字段	说明
YDOrder	YDOrderFlag	固定为YD_YOF_OptionExecuteTogether

参数	字段	说明
	OrderLocalID	柜台本地编号。供易达柜台内部使用，其正负性、唯一性、递增性等特征可能会因为交易所、会员、易达交易系统的版本而发生变化。请勿使用该字段作为报单的标识。
	OrderSysID	交易所返回的系统报单号，若交易所报单编号超过OrderSysID的最大长度，则会截取部分
	LongOrderSysID	交易所返回的系统报单号，不会截取交易所报单编号，保留全精度
	InsertTime	从该交易日开始（17点整）到报单时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	InsertTimeStamp	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	OrderStatus	申请成功为YD_OS_Queuing，申请失败则为YD_OS_Rejected
	ErrorNo	成功时为0，错误时为交易所错误号
YDInstrument	pInstrument	待合并行权的期权合约指针
YDInstrument	pInstrument2	待合并行权的期权合约指针2
YDAccount	pAccount	当前API登录账户

由于期权合并行权申请成功后，申请委托处于挂单状态，因此，如果要撤销行权申请，可以使用cancelOrder撤销。撤销期权合并行权申请和撤销普通报单的用法以及回报是相同的，具体请参考撤单。

7.7 转托管

易达支持深交所的转托管业务，将投资者的所有持仓全部转移到其他PBU上，通常会在投资者从易达柜台转出时调用，只能由管理端调用。

转托管使用insertOrder发起，参数说明如下表所示。

参数	字段	说明
YDInputOrder	YDOrderFlag	填写YD_YOF_Designation
	HedgeFlag	填写YD_HF_Speculation
	TransfereePBUID	转托管目标PBU
YDInstrument	pInstrument	任意深交所证券指针
YDAccount	pAccount	填写转托管的投资者指针

7.8 尚未支持的业务

易达的目标是在保持性能的情况下尽可能支持投资者所需要的业务。基于性能和使用易达产品的投资者的实际业务需求考虑，易达柜台尚未支持以下交易业务。

交易所	不支持业务
上期所、能源所	结算价交易TAS指令

如果您对上述业务有实际需求的，请通过经纪商联系易达，假若该需求确实存在普遍性的，易达会努力满足您的业务需要。

7.9 交易限制

7.9.1 交易权限

出于风险管理和投资者适当性的需要，经纪商有在投资者、交易所、产品和合约层面设置交易权限的需求；投资者也有主动设置部分合约交易权限的风控需求。因此，易达提供了四个交易权限设置来源以满足不同的风控设置需求：

- 管理员设置的永久权限：设置后永久有效。只有管理员能设置该来源，投资者无法设置。
- 投资者设置的永久权限：设置后永久有效。管理员和投资者都可设置。
- 管理员设置的临时权限：设置后当前交易日内有效。只有管理员能设置该来源，投资者无法设置。目前由事后风控触发规则后设置，以确保事后风控规则只有在当前交易日有效。
- 投资者设置的临时权限：设置后当前交易日内有效。管理员和投资者都可设置。

> 无论合约的交易权限如何，撤单是始终可以执行的。

虽然易达允许在交易权限设置源分别设置，但投资者在具体合约上的权限是汇总上述四个来源后的结果。投资者设置交易权限的方法如下所示：

```
virtual bool setTradingRight(const YDAccount *pAccount,const YDInstrument *pInstrument,const YDProduct *pProduct,const YDExchange *pExchange,int tradingRight,int requestID=0,int tradingRightSource=YD_TRS_AdminPermanent)
```

上述方法中各参数的说明如下：

参数	说明
pAccount	指向账户的指针。当前投资者的账户指针可以通过getMyAccount获取。
pInstrument	合约指针，表示设置该合约的权限。此时请设置pProduct和pExchange为NULL
pProduct	产品指针，表示设置该产品的权限。此时请设置pInstrument和pExchange为NULL
pExchange	交易所指针，表示设置该交易所的权限。此时请设置pProduct和pInstrument为NULL
tradingRight	需要设置的权限。可选项为： YD_TR_Allow=0：允许所有交易。最为宽松的设置。 YD_TR_CloseOnly=1：仅允许平仓 YD_TR_Forbidden=2：不允许任何交易。最为严格的设置。
requestID	用于区分不同的设置请求，通常设置0即可。可以在notifyResponse中接收到请求的requestID。

参数	说明
tradingRightSource	交易权限设置来源 YD_TRS_AdminPermanent=0：管理员设置的永久权限 YD_TRS_UserPermanent=1：投资者设置的永久权限 YD_TRS_AdminTemp=2：管理员设置的临时权限 YD_TRS_UserTemp=3：投资者设置的临时权限

投资者在具体合约上的最终权限取决于该投资者在该合约所属交易所、该合约所属产品以及该合约上最严格的设置，伪代码表述如下：

```
int getInstrumentTradingRight(YDApi *api, YDInstrument *instrument)
{
    accountInfo=api->getMyAccount();
        accountExchangeInfo=api->getAccountExchangeInfo(instrument->m_pExchange);
    accountProductInfo=api->getAccountProductInfo(instrument->m_pProduct);
        accountInstrumentInfo=api->getAccountInstrumentInfo(instrument);

    accountInfo->TradingRight = max(
        accountInfo->TradingRightFromSource[YD_TRS_AdminPermanent],
        accountInfo->TradingRightFromSource[YD_TRS_UserPermanent],
            accountInfo->TradingRightFromSource[YD_TRS_AdminTemp],
        accountInfo->TradingRightFromSource[YD_TRS_UserTemp]
    );

    accountExchangeInfo->TradingRight = max(
        accountExchangeInfo->TradingRightFromSource[YD_TRS_AdminPermanent],
        accountExchangeInfo->TradingRightFromSource[YD_TRS_UserPermanent],
            accountExchangeInfo->TradingRightFromSource[YD_TRS_AdminTemp],
        accountExchangeInfo->TradingRightFromSource[YD_TRS_UserTemp]
    );

    accountProductInfo->TradingRight = max(
        accountProductInfo->TradingRightFromSource[YD_TRS_AdminPermanent],
        accountProductInfo->TradingRightFromSource[YD_TRS_UserPermanent],
            accountProductInfo->TradingRightFromSource[YD_TRS_AdminTemp],
        accountProductInfo->TradingRightFromSource[YD_TRS_UserTemp]
    );

    accountInstrumentInfo->TradingRight = max(
        accountInstrumentInfo->TradingRightFromSource[YD_TRS_AdminPermanent],
        accountInstrumentInfo->TradingRightFromSource[YD_TRS_UserPermanent],
            accountInstrumentInfo->TradingRightFromSource[YD_TRS_AdminTemp],
        accountInstrumentInfo->TradingRightFromSource[YD_TRS_UserTemp]
    );

        return max(
        accountInfo->TradingRight,
        accountExchangeInfo->TradingRight,
            accountProductInfo->TradingRight,
        accountInstrumentInfo->TradingRight
    );
}
```

盘中若交易权限发生改变，通常是由于管理员主动设置或者事后风控触发，可以通过以下回调函数获得权限变更通知。收到任何下面三个回调中的一个时，都应该使用上面的伪代码函数getInstrumentTradingRight获取您关心的合约的交易权限：

```
virtual void notifyAccount(const YDAccount *pAccount)
virtual void notifyAccountExchangeInfo(const YDAccountExchangeInfo *pAccountExchangeInfo)
virtual void notifyAccountProductInfo(const YDAccountProductInfo *pAccountProductInfo)
virtual void notifyAccountInstrumentInfo(const YDAccountInstrumentInfo
*pAccountInstrumentInfo)
```

如果尝试在没有交易权限的合约上开平仓，会收到错单回报notifyOrder或者错误报价回报notifyQuote，其中的ErrorNo为YD_ERROR_NoTradingRight=10。

组合合约的交易权限并非使用设置在该组合合约上的权限，而是分别判断组合合约的两腿合约上设置的权限是否满足交要求。假设某组合合约的左右两腿分别为A、B合约，根据投资者在两个合约上设置的权限不同，不同的组合合约交易如下：

- A、B合约为允许交易，那么该组合合约所有交易都被允许
- A合约为允许交易，B合约为只能平仓，那么该组合合约只有同时平两腿和开左腿平右腿两种交易被允许
- A、B合约为只能平仓，那么该组合合约只有同时平两腿的交易被允许
- A合约为允许交易，B合约为禁止交易，那么该组合合约所有交易都被禁止

7.9.2 交易约束

现货业务中证券公司会约束投资者在交易和非交易上行为，例如禁止买入、禁止卖出、禁止转托管等。交易方面，交易约束可以仅禁止卖出而允许买入，而交易权限则无法表示；非交易方面，交易权限机制则完全无法适用。因此，易达增加了交易约束功能，用于满足证券公司设置交易约束并在盘中修改的业务需要。

调用adjustCashTradingConstraint后，会通过notifyResponse返回是否成功，若成功则还会收到notifyAdjustCashTradingConstraint回调，若失败则不会收到。该设置方法只对现货柜台有效，且只能由拥有设置交易权限的管理员调用设置，设置后当前交易日有效。

```
// 设置交易约束
virtual bool adjustCashTradingConstraint(const YDAdjustCashTradingConstraint
*pAdjustCashTradingConstraint,int requestID=0)

// 交易约束设置回报
virtual void notifyAdjustCashTradingConstraint(const YDAdjustCashTradingConstraint
*pAdjustCashTradingConstraint)
```

YDAdjustCashTradingConstraint的结构如下：

字段	说明
AccountRef	设置交易约束的资金账户序号，必须设置
ExchangeRef	交易所序号。不设置表示不限制被设置的合约所属交易所
ProductRef	产品序号。不设置表示不限制被设置的合约所属产品
InstrumentRef	合约序号。不设置表示不限制被设置的合约
IsSetting	>0表示设置约束，0表示取消约束
CashTradingConstraint	现货交易约束位图，各位表示的含义如下： 0：禁止买入 1：禁止卖出

ExchangeRef、ProductRef、InstrumentRef共同框定了被设置交易约束的合约的范围，例如：

- 仅设置了ExchangeRef，表示所有属于该交易所的合约全部被设置交易约束
- 仅设置了ProductRef，表示所有属于该产品的合约全部被设置交易约束

- 仅设置了InstrumentRef，表示只有该合约被设置交易约束
- 如果ExchangeRef、ProductRef、InstrumentRef都没有被设置，则表示所有合约都被设置交易约束

通常而言设置了InstrumentRef后就不必设置ExchangeRef和ProductRef，设置了ProductRef后就不比设置ExchangeRef，若设置了细粒度范围后又设置了粗粒度范围，则需要保证他们具有逻辑一致性，否则将会导致设置失败。例如，ExchangeRef设置上交所，但是InstrumentRef设置了深交所的合约即为失败的设置。

当框定设置范围后，针对每一个合约的设置方式依赖于IsSetting的值。假设某投资者在某合约上当前的交易约束为禁止卖出但允许买入（投资者当前的交易约束设置可以从YDExtendedAccountInstrumentInfo.CashTradingConstraint获取），即0b10，下面演示不同设置方法的效果：

- IsSetting=1，CashTradingConstraint=0b1或0b11，若设置成功，则该投资者的交易约束为0b11
- IsSetting=0，CashTradingConstraint=0b10或者0b11，若设置成功，则该投资者的交易约束为0b0

7.9.3 报撤单笔数限制

默认情况下，易达柜台并不控制投资者的报单和撤单笔数。但由于易达柜台的内存数据库表存在容量上限，一旦某内存数据库表达到其容量上限，整个易达柜台就会处于不可用状态。为避免某些投资者异常交易导致的报撤单笔数暴增而影响到柜台的其他客户，经纪商可以限制每个投资者的报单笔数和撤单笔数。

报单笔数和撤单笔数控制是在资金账户层面的，如果投资者建立了多个连接，所有连接的报撤单笔数是汇总计算的。只有发送到交易所并收到交易所回报的普通报单会增加投资者的报单笔数，报价、撤单、衍生报单、柜台返回的错误报单都不会计入报单笔数。柜台接收到的所有撤单请求都会增加投资自的撤单笔数，撤报价不会计入撤单笔数。

> 当前版本中，易达柜台内存数据库报单、成交的上限为1677万笔，报价的上限为838万笔，如业务发生变化，正常业务需要更大的空间，易达随时可以扩容。

报单笔数限制可在YDAccount.MaxOrderCount中查到，撤单笔数限制可在YDAccount.MaxCancelCount中查到。如果该值为-1，则表示不限制，默认为不限制。

对于使用ydExtendedApi的投资者，可以在YDExtendedAccount.UsedOrderCount查询到当前账户的总报单笔数。如果投资者使用该资金账户建立了多个连接，YDExtendedAccount.UserdOrderCount包含了所有连接的报单笔数。投资者的总撤单笔数只在柜台端记录，无法通过API查看；同时，柜台重启后撤单表会被清空，因此投资者的总撤单笔数也会因为重启被清零。

在投资者达到该账户的报单笔数上限后的下一张报单会收到notifyOrder返回的YD_ERROR_TooManyOrders=36错误；类似的，在投资者达到该账户的撤单笔数上限后的下一张报单会收到notifyFailedCancelOrder返回的YD_ERROR_TooManyCancels=89错误。为保护柜台表空间不被超量非正常报单挤满，如果该投资者继续报单将直接被柜台丢弃，不会有任何错误回报返回。请投资者务必检查YD_ERROR_TooManyOrders=36和YD_ERROR_TooManyCancels=89错误并在收到该错误后停止或者调整报撤单逻辑，如有疑问请联系经纪商核实。

如果柜台内存数据库资源耗尽，投资者的任意报单都会收到notifyOrder返回的 YD_ERROR_MemoryExceed=7错误。

7.9.4 登录数限制

默认情况下，易达柜台不会限制资金账户的登录连接个数，投资者可以使用同一个账户无限制的登录易达柜台。但是，过多的连接会造成TCP下行线程始终忙于下发下行流水，这对易达柜台的高效运行是有害的。为了避免连接数过多而影响柜台的性能，经纪商可以为每个投资者设置登录数限制。

登录数总数可以在YDAccount.MaxLoginCount中查到，如果该值为-1，则表示不限制，默认为不限制。当前登录数可以在YDAccount.LoginCount中查到。

在投资者达到该账户的登录数上限后，继续登录会在notifyLogin中返回YD_ERROR_TooManyRequests=20错误。

如果了解更多多连接的情况，请参阅多连接相关内容。

7.9.5 自成交检查

为避免因投资者自成交被交易所限制交易，易达柜台提供了自成交检查功能。易达的自成交检查功能做了性能优化，因此不用因为担心性能问题而关闭自成交检查。

该功能默认是打开的，如果投资者在策略程序中已经做了检查，可以请经纪商帮助关闭该账户的自成交检查功能。投资者可以检查YDAccount.AccountFlag中是否被设置了YD_AF_DisableSelfTradeCheck标志，如果被设置则表示该账户的自成交检查是关闭的。

对于新报单，柜台会检查其与处于未完成状态的报价和特定报单之间是否存在价格重叠；对于新报价，除中金所之外，柜台会检查其与处于未完成状态的特定报单之间是否存在价格重叠。例如，若某合约当前有卖单价格10元，若此时想要报入11元的买单，那么就会被系统判定为自成交，而如果报入9元的买单，则不会被判定为自成交。上文所指的特定报单在不同交易所含义不同，具体如下：

- 中金所国债期货的限价、FAK和FOK报单；中金所其他产品的限价单
- 其他交易所的限价单

上期所、能源所、大商所、郑商所、广期所在集合竞价时豁免自成，因此，上述交易所在集合竞价阶段的报单不会被易达柜台判定为自成交。

> 如果在中金所交易时遇到了"1138 订单触发自成交"的错误，说明所在经纪商已经向中金所申请开通了"自成交防范功能"，该功能会检查同一交易编码下的FAK/FOK单与限价单是否发生价格重叠，若重叠则会被交易所拦截。自成交防范功能是中金所向经纪商提供的一种保护功能，启用后对该经纪商所有交易编码生效，不可以在投资者维度单独设置。虽然自成交防范功能拦截了FAK/FOK的自成交行为，但不代表FAK/FOK单引起的自成交会被中金所监管部门判定为自成交。在《中国金融期货交易所异常交易管理办法》股指期权有关监管标准及处理程序中明确指出，在统计客户自成交、频繁报撤单和大额报撤单次数时，因即时全部成交或撤销限价指令、即时成交剩余撤销限价指令、市价指令形成的自成交行为不计入自成交次数统计。

虽然某些交易所给予投资者一定的自成交豁免额度（例如每个合约允许5次自成交），但是考虑到自成交检查是基于投资者交易编码的，如果同一个投资者的交易编码在多家经纪商交易造成自成交的，也会被交易所计算在自成交范围内，而此类自成交控制较为困难。出于保护投资者的角度考虑，为了把豁免额留给不太可控的同交易编码跨经纪商自成交，易达柜台不允许任何潜在的自成交报单，所有被系统检测为可能构成自成交的都会被柜台直接拒绝。

任何被判定为自成交的报单，都会通过notifyOrder返回的YD_ERROR_PossibleSelfTrade=25错误。

7.9.5.1 中金所报价自成交检查

由于中金所支持多层次报价，自成交判断会变得复杂，为确保不会产生自成交，当顶指定单时，被顶报价会被跳过检查自成交，其他报价和特定报单仍然会被检查是否与新报价产生自成交；当顶最后一单时，由于无法知晓交易所收到的最后一单，因此易达从严检查，即会检查所有报价和特定报单是否与新报价产生自成交。

易达柜台无法知晓交易所收到的最后一单会有多种原因构成，包括但不限于：

- 投资者在两套柜台中同时发送报价
- 投资者在一套柜台中连续发单但柜台收到的顺序和发向交易所的顺序不一致。例如连续发普通报价A、顶最后一单B、普通报价C，A和B构成自成交，如果按照顶最后一单规则，A被顶所以可以被跳过检查，B能正常发出，C经检查不与A和B构成自成交也能正常发出。但是由于系统内部调度或堵单等原因导致最终发向交易所的顺序为A/C/B，此时交易所收到B时的最后一单是C，因此C会被顶单，从而A和B构成了自成交。

上述原因虽然理论上成立，但是现实交易中却不太会发生。通常投资者不会在两套柜台交易同一个合约，也不会混用普通报价和顶最后一单，一般股指做市商不会建立多层次报价，所有报价均会带上顶最后一单标记，可以证明此时不会发生任何潜在自成交问题，而国债衍生品做市商通常会建立多层次报价，通常只会使用普通报价和顶指定单，此时也不会有潜在自成交问题。因此，当投资者认为自己报价行为中不存在潜在自成交问题时，可以请经纪商启用该投资者的中金所报价宽松自成交检查功能，打开该功能后，中金所顶最后一单报价不再与其他报价检查是否构成自成交，但仍需检查与普通报单是否构成自成交，这与其他支持顶单的交易所检查自成交的逻辑一致。报价宽松自成交检查功能不影响顶指定单的自成交检查逻辑。

投资者可通过YDAccount.AccountFlag检查YD_AF_RelaxSelfTradeCheckForQuote标志以判断是否启用了该功能。

7.9.6 实控账户组

按照监管规定，交易所在执行持仓限额、交易限额、异常交易行为（自成交量、报撤单笔数、大额报撤单笔数、日内开仓交易量等）管理等制度和规定时，对一组实际控制关系账户的交易、持仓等合并计算。为帮助经纪商和投资者切实履行实控账户组的监管义务，从1.386版本起易达柜台支持将同一套柜台中的多个资金账户设置为实控账户组。实控账户组需要经纪商根据监管要求或者在投资者要求下主动设置，否则易达柜台不会认定任何账户的实控关系。

易达支持基于实控账户组的自成交控制，柜台将实控账户组成员的所有报单和报价视作一个账户予以控制，控制逻辑请参考自成交检查。若实控账户组内的成员账户关闭了自成交检查功能，那么该成员账户的报单和报价不会参与账户组以及该账户自身的自成交检查。

由于可以通过单账户设置额度的方式变相实现实控账户组的总额控制，其他监管限制未在实控账户组层面支持，如确有支持需求的，请与我们联系。

实控账户组的成员账户信息不会下发到客户端，仅在保存在柜台本地，投资者可以向经纪商咨询具体设置情况。

7.9.7 报单号单调递增检查

在过去的API版本中，报单和报价等上行指令中的OrderRef是没有递增要求的，投资者可以根据需要随意设置OrderRef。易达柜台在生产中曾经发生过因为网络原因，柜台UDP接口收到大量重复报单，导致投资者交易受到了影响。

为避免发生此类问题，建议投资者使用非0报单组进行报单，利用其检查功能规避重复报单。详情请参见报单组。

> 在过去版本的API中，为解决上述问题，易达为每个在管理端新开立的资金账户默认打开了报单号检查功能。柜台将检查打开该功能的投资者的OrderRef是否单调递增的，若收到了OrderRef非单调递增的报单，则将返回ErrorNo为YD_ERROR_InvalidOrderRef=78的错误回报。
>
> 报单号单调递增检查的范围仅限于API实例的范围内，即同一个API实例发出的报单要求单调递增，但是相同账户创建的多个API实例之间的报单的OrderRef没有任何限制，因此，多连接场景下，使用OrderRef后几位作为SessionID的做法仍然是生效的。裸协议报单是作为一个单独的源进行检查的，不会和API的UDP报单相冲突。若柜台发生了重启（如日盘启动）而客户端因为开启了客户端高可用而没有重启的，报单号可以重头开始。
>
> 投资者可以检查YDAccount.AccountFlag是否设置了YD_AF_OrderRefCheck标志了解该功能的启用情况，若设置了该标志则表示报单号单调递增检查功能开启。
>
> 为避免不必要的重复检查，建议对于已经使用非0报单组进行报单的投资者关闭上述报单号检查功能。若投资者继续使用编号为0的报单组进行报单同时不希望报单号检查影响现有OrderRef的编码逻辑，也可以请经纪商关闭账号上的报单检查开关。

7.9.8 柜台流控

为防止部分投资者密集报单引发席位流控从而影响到同柜台上其他投资者的正常交易，经纪商可对投资者设置柜台流控，控制投资者每秒向柜台发送指令的速度。同一个资金账户多个登录连接加总计算后统一流控。投资者向柜台发送的以下指令会被计入流控，包括报撤单、询价、组合与解组合、行权履约指令，无论指令是否被柜台拒绝均会被计入流控。批量报撤单指令中包含的每一个报撤单独立计入柜台流控。以下两个场景不受柜台流控影响：

- 管理员可以投资者名义在柜台交易，包括极端市场时强平强减，或者投资者遇到技术故障时管理员帮助投资者平仓或者撤单等诸多场景，为了尽快释放风险，管理员的交易指令不计入柜台流控
- 考虑到做市商柜台通常为独占柜台，因此，做市商的报价、撤报价指令不计入柜台流控

投资者可以从YDAccount.MaxRequestSpeed中获取柜台流控阈值，若阈值被设置为-1，则表示无柜台流控。当投资者的指令被柜台流控时，回报中的ErrorNo为YD_ERROR_AccountRequestTooFast=24。

7.10 查询交易信息

ydApi并不保存流水，因此下列查询功能只能在ydExtendedApi中使用。

查询语句是慢速调用，不应在交易的线程中调用，也尽可能不要频繁调用查询语句，以免造成系统卡顿。

7.10.1 查询报单

```
/// getOrder by orderSysID can only be used for orders have been accepted by exchange
virtual const YDExtendedOrder *getOrder(int orderSysID,const YDExchange *pExchange,int YDOrderFlag=YD_YOF_Normal)
virtual const YDExtendedOrder *getOrder(long long longOrderSysID,const YDExchange *pExchange,int YDOrderFlag=YD_YOF_Normal)
```

上面的方法可以用来查询**已经收到交易所回报的报单**或者报价衍生报单，所有通过insertOrder系列报单方法报入的报单都可以使用本函数查询到，默认情况下查询普通报单，可以通过设置YDOrderFlag查询其他类型的报单。

```
/// getOrder by orderRef can only be used for orders using checkAndInsertOrder
virtual const YDExtendedOrder *getOrder(int orderRef,unsigned orderGroupID=0,const YDAccount *pAccount=NULL)

/// getQuoteDerivedOrder can only be used for orders derived oder by using checkAndInsertQuote
virtual const YDExtendedOrder *getQuoteDerivedOrder(int orderRef,int direction,unsigned orderGroupID=0,const YDAccount *pAccount=NULL)
```

对于使用ydExtendedApi的checkAndInsertOrder报入的报单可以使用上述两个方法获取，只要checkAndInsertOrder和checkAndInsertQuote调用完成，就能通过这两个方法查询到，而无需收到交易所报单回报。其中getOrder可以查询到除了报价衍生报单之外的报单，而getQuoteDerivedOrder可以查询报价衍生报单，相比getOrder增加了报单方向direction的参数用于区分双边报价中的买卖衍生报单。

```
/// orders must have spaces of count, return real number of orders(may be greater than count). Only partial will be set if no enough space. Only orders accepted by exchange can be found in this function
virtual unsigned findOrders(const YDOrderFilter *pFilter,unsigned count,const YDExtendedOrder *orders[])

/// User should call destroy method of return object to free memory after using following method
virtual YDQueryResult<YDExtendedOrder> *findOrders(const YDOrderFilter *pFilter)
virtual YDQueryResult<YDExtendedOrder> *findPendingOrders(const YDOrderFilter *pFilter)
```

易达提供了上述两类共三种不同的批量查询**已经收到交易所回报的报单**的方法，其中findOrders查找所有符合条件的报单，findPendingOrders查找所有符合条件的挂单委托。各方法查询条件的使用方法是相同的，判断某报单是否符合查询条件的伪代码逻辑如下所示：

```
if YDOrder.OrderSysID < 0
    return false

if YDOrder.YDOrderFlag not in YDOrderFilter.YDOrderFlags
    return false

if YDOrderFilter.StartTime >= 0 and YDOrder.InsertTime > YDOrderFilter.StartTime
    return false

if YDOrderFilter.EndTime >= 0 and YDOrder.InsertTime < YDOrderFilter.EndTime
    return false

if YDOrderFilter.pExchange != NULL and YDOrderFilter.pExchange != YDOrder.pExchange
    return false

if YDOrder.YDOrderFlag ==  YD_YOF_CombPosition
```

```
    if YDOrderFilter.pCombPositionDef != NULL and YDOrderFilter.pCombPositionDef != YDOrder.pCombPositionDef
        return false
else
    if YDOrderFilter.pInstrument != NULL and YDOrderFilter.pInstrument != YDOrder.pInstrument
        return false
    if YDOrderFilter.pProduct != NULL and YDOrderFilter.pProduct != YDOrder.pProduct
        return false
return true
```

各个字段填写说明如下：

参数	字段	说明
YDOrderFilter	StartTime	以TimeID形式表示的起始时间，-1代表无限制。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	EndTime	以TimeID形式表示的结束时间，-1代表无限制。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	YDOrderFlags	设置要查询的YDOrderFlag的标志位，可以设置多个。 例如，要查询普通报单和组合报单，那么可以使用以下表达式设置： 1<<YD_YOF_Normal | 1<<YD_YOF_CombPosition
	pCombPositionDef	传统组合持仓定义。对于YD_YOF_CombPosition的报单只检查本参数和pExchange。如设置为NULL则不限制。
	pInstrument	合约指针，对于非YD_YOF_CombPosition的报单生效。如设置为NULL则不限制。
	pProduct	产品指针，对于非YD_YOF_CombPosition的报单生效。如设置为NULL则不限制。
	pExchange	交易所指针。如设置为NULL则不限制。
	pAccount	投资者请始终设置为NULL

第一种方法需要投资者事先分配固定长度的YDExtendedOrder指针数组，如果预分配的长度不足以容纳的，只会填入预分配数组长度的报单。无论预分配长度是否足够，本方法的返回值都是返回满足查询条件的报单的总数，投资者可以利用这个特点，可以调用findOrders(pFilter, 0, NULL)来快速获取满足条件的报单的总数。

- 本方法的好处是在报单不多而且预分配数组长度足够时，可以重复利用预分配的指针数组，而不用每次查询都分配
- 本方法的缺点是如果报单较多时，可能需要两次调用（第一次获取总报单数，第二次分配可容纳所有报单的数组后再次调用）才能完整获取所有满足条件的报单

第二种方法可以帮助投资者分配能容纳所有满足条件的报单的空间，但是需要投资者在使用完成后主动调用YDQueryResult.destroy()销毁分配的空间。相比起第一种方法：

- 本方法的好处是调用一次就返回所有的报单

- 本方法的缺点是需要投资者主动释放由本方法分配的空间，而且每次调用都会分配新的空间，频繁的分配与释放对缓存非常不友好。

7.10.2 查询成交

```
/// trades must have spaces of count, return real number of trades(may be greater than count). Only partial will be set if no enough space
virtual unsigned findTrades(const YDTradeFilter *pFilter,unsigned count,const YDExtendedTrade *trades[])

/// User should call destroy method of return object to free memory after using following method
virtual YDQueryResult<YDExtendedTrade> *findTrades(const YDTradeFilter *pFilter)
```

批量查询成交的方法与查询报单类似，具体内容可以参考查询报单的相关内容。判断某成交是否符合查询条件的伪代码逻辑如下所示：

```
if YDTradeFilter.StartTime >= 0 and YDTrade.InsertTime > YDTradeFilter.StartTime
    return false

if YDTradeFilter.EndTime >= 0 and YDTrade.InsertTime < YDTradeFilter.EndTime
    return false

if YDTradeFilter.pInstrument != NULL and YDTradeFilter.pInstrument != YDTrade.pInstrument
    return false

if YDTradeFilter.pProduct != NULL and YDTradeFilter.pProduct != YDTrade.pProduct
    return false

if YDTradeFilter.pExchange != NULL and YDTradeFilter.pExchange != YDTrade.pExchange
    return false

return true
```

各个字段填写说明如下：

参数	字段	说明
YDTradeFilter	StartTime	以TimeID形式表示的起始时间，-1代表无限制。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	EndTime	以TimeID形式表示的结束时间，-1代表无限制。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	pInstrument	合约指针。如设置为NULL则不限制。
	pProduct	产品指针。如设置为NULL则不限制。
	pExchange	交易所指针。如设置为NULL则不限制。
	pAccount	投资者请始终设置为NULL

7.10.3 查询报价

```
/// getQuote by quoteSysID can only be used for quotes have been accepted by exchange
virtual const YDExtendedQuote *getQuote(int quoteSysID,const YDExchange *pExchange)
virtual const YDExtendedQuote *getQuote(long long longQuoteSysID,const YDExchange *pExchange)
```

上面的方法可以用来查询**已经收到交易所回报的报价**，所有通过insertQuote系列报价方法报入的报价都可以使用本函数查询到。

```
/// getQuote by orderRef can only be used for quotes using checkAndInsertQuote
virtual const YDExtendedQuote *getQuote(int orderRef,unsigned orderGroupID=0,const YDAccount *pAccount=NULL)
```

对于使用ydExtendedApi的checkAndInsertQuote报入的报价可以使用上述方法获取，只要checkAndInsertQuote调用完成，就能通过这两个方法查询到，而无需收到交易所报单回报。

```
/// quotes must have spaces of count, return real number of quotes(may be greater than count). Only partial will be set if no enough space. Only quotes accepted by exchange can be found in this function
virtual unsigned findQuotes(const YDQuoteFilter *pFilter,unsigned count,const YDExtendedQuote *quotes[])

/// User should call destroy method of return object to free memory after using following method
virtual YDQueryResult<YDExtendedQuote> *findQuotes(const YDQuoteFilter *pFilter)
virtual YDQueryResult<YDExtendedQuote> *findPendingQuotes(const YDQuoteFilter *pFilter)
```

批量查询报价的方法与查询报单类似，具体内容可以参考查询报单的相关内容。判断某报价是否符合查询条件的伪代码逻辑如下所示：

```
if YDQuote.OrderSysID < 0
    return false

if YDQuoteFilter.StartTime >= 0 and (any DerivedOrder of YDQuote).InsertTime > YDQuoteFilter.StartTime
    return false

if YDQuoteFilter.EndTime >= 0 and (any DerivedOrder of YDQuote).InsertTime < YDQuoteFilter.EndTime
    return false

if YDQuoteFilter.pInstrument != NULL and YDQuoteFilter.pInstrument != YDQuote.pInstrument
    return false

if YDQuoteFilter.pProduct != NULL and YDQuoteFilter.pProduct != YDQuote.pProduct
    return false

if YDQuoteFilter.pExchange != NULL and YDQuoteFilter.pExchange != YDQuote.pExchange
    return false

return true
```

各个字段填写说明如下：

参数	字段	说明
YDTradeFilter	StartTime	以TimeID形式表示的起始时间，-1代表无限制。报价的任意衍生报单的时间大于开始时间就符合条件。 请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	EndTime	以TimeID形式表示的结束时间，-1代表无限制。报价的任意衍生报单的时间小于结束时间就符合条件。 请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 TimeID和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
	pInstrument	合约指针。如设置为NULL则不限制。
	pProduct	产品指针。如设置为NULL则不限制。
	pExchange	交易所指针。如设置为NULL则不限制。
	pAccount	投资者请始终设置为NULL

7.11 交易阶段

易达支持发送汇总和明细两种交易阶段通告：

- 汇总交易阶段通告是始终发送的交易阶段通告。为了减少通讯量和对API的冲击，汇总交易阶段通告仅通知客户端交易所发布状态变化的时刻，若同一时刻有多个事件发生，只有第一个会发出，即相同时刻的事件只会推送第一个。
- 明细交易阶段通告是默认不发送的交易交易阶段通告。由于某些交易所的交易阶段通告是基于合约的，导致在同一时间会有大量的通告向所有投资者发送，如果此时有报单回报则会被拥塞。因此，请谨慎使用明细交易阶段通告。一旦本功能被打开，则该柜台上所有投资者都将收到明细交易阶段通告。可以通过SystemParam中的TradingSegmentDetail查看柜台是否启用该功能，详情参见系统参数。

7.11.1 汇总交易阶段

汇总交易阶段通过以下回调函数返回。segmentTime表示从该交易日开始到此时间的秒数，其编码方式请参见报单回报中InsertTime的相关说明。

```
virtual void notifyTradingSegment(const YDExchange *pExchange, int segmentTime)
```

下面列出了目前各个交易所的交易状态变化时间供参考，若交易所有变化恕不另行通知，请以生产中接收到的实际时间为准。

交易所	日盘	夜盘
中金所	09:25:00 09:29:00 09:30:00 11:30:00 13:00:00 14:57:00 15:00:00 15:15:00	
能源所	09:00:00 10:15:00 10:30:00 11:30:00 13:30:00 15:00:00	20:55:00 20:59:00 21:00:00 23:00:00 01:00:00 02:30:00
上期所	08:55:00 08:59:00 09:00:00 10:15:00 10:30:00 11:30:00 13:30:00 15:00:00	20:55:00 20:59:00 21:00:00 23:00:00 01:00:00 02:30:00
大商所	08:54:50 08:55:00 08:58:50 08:59:00 08:59:50 09:00:00 10:14:50 10:15:00 10:29:50 10:30:00 11:29:50 11:30:00 13:29:50 13:30:00 14:59:50 15:00:00	20:54:50 20:55:00 20:58:50 20:59:00 20:59:50 21:00:00 22:59:50 23:00:00

7.11.2 明细交易阶段

明细交易阶段通过以下回调函数返回。交易阶段的返回维度各个交易所不尽相同，可能在交易所、产品或合约共三个维度上返回明细交易阶段，每次明细交易阶段通告只可能上述三个维度中的一个。明细交易阶段会在notifyFinishInit之后向投资者回调，但投资者不应假设一定在notifyCaughtUp之前能收齐所有交易所、产品或合约的状态，可能会有部分明细交易阶段是在notifyCaughtUp之后才能收到，策略程序在没有收到明细交易阶段之前，应默认该合约的明细交易阶段为非交易，直到收到与该合约相关的明细交易阶段通知。易达柜台会确保所有合约的交易状态都将被正确发送，无论柜台是否在盘中被重启过。

> 造成notifyCaughtUp之前无法收齐的原因主要有两个，一是当前可能处于盘前状态，交易所并没有推送任何明细交易阶段；二是由于易达流水发送机制，如果在发送流水时有更新的明细交易阶段，则会跳过旧版本的明细交易阶段，如果新版本的明细交易的流水号阶段大于API登录时刻的最大报单编号，那么该明细交易阶段的记录将会延后至notifyCaughtUp之后发送。

```
virtual void notifyTradingSegmentDetail(const YDTradingSegmentDetail *pTradingSegmentDetail)
```

返回的YDTradingSegmentDetail的字段说明如下。

字段	说明
ExchangeRef	交易所序号。返回交易所明细交易阶段时有意义，否则为-1。
ProductRef	产品序号。返回产品明细交易阶段时有意义，否则为-1。
InstrumentRef	合约序号。返回合约明细交易阶段时有意义，否则为-1。
m_pExchange	交易所指针。返回交易所明细交易阶段时有意义，否则为NULL。
m_pProduct	产品指针。返回产品明细交易阶段时有意义，否则为NULL。
m_pInstrument	合约指针。返回合约明细交易阶段时有意义，否则为NULL。
SegmentTime	从该交易日开始到此时间的秒数。其编码方式请参见报单回报中InsertTime的相关说明。
TradingStatus	交易阶段状态： YD_TS_NoTrading：非交易 YD_TS_Continuous：连续交易 YD_TS_Auction：集合竞价

7.12 报单组

为了更好的支持投资者对区分报单和报单号单调递增检查的需求，易达在`1.280`版本中增加了报单组功能。投资者可以在YDInputOrder和YDInputQuote中设置OrderGroupID和GroupOrderRefControl使用报单组功能。

> **Important**
>
> 1.280版本引入报单组功能时支持最大63个报单组，1.486版本将报单组总量从63增加到255个。当使用1.486版本前的API连接1.486版本的柜台时，仍然只能使用63个报单组。要使用最大255个报单组则必须将API和柜台都升级到1.486及以后版本。

字段	说明
OrderGroupID	指定报单和报价所属的逻辑组，可以用于区分报单所属的策略、连接以及其他逻辑分组 投资者可以使用0-255号报单组。 为保兼容，0号报单组不检查OrderRef单调递增性，1-255号报单组将按设置检查OrderRef单调性
GroupOrderRefControl	指定投资者要求柜台对报单OrderRef的单调递增性检查方式： YD_GORF_Increase：OrderRef单调递增，前后两单OrderRef的间隔必须大于等于1 YD_GORF_IncreaseOne：OrderRef严格单调递增，前后两单OrderRef的间隔必须为1

在报单和报价上指定OrderGroupID和GroupOrderRefControl后，柜台将检查报单所属报单组的单调性。若检查失败，则直接通过notifyOrder返回ErrorNo=YD_ERROR_InvalidGroupOrderRef的错单，并在该错单的MaxOrderRef中返回报单中设置的OrderGroupID当前最大OrderRef。无论报单成功与失败，YDInputOrder和YDInputQuote上设置的OrderGroupID和GroupOrderRefControl都将通过YDOrder和YDQuote带回。

通过在每张报单上指定OrderGroupID和GroupOrderRefControl，给予了投资者最大的使用便利。对属于同一OrderGroupID的报单，投资者可以对第一张设置YD_GORF_Increase实现单调递增检查，第二张设置YD_GORF_IncreaseOne实现严格单调递增检查。

在首次登录成功或者断线重连并登录成功的回调函数notifyLogin之后，柜台将通过以下回调函数返回每个OrderGroupID当前最大OrderRef。请务必记录该回调中的返回值，并在后续报单中使用符合递增规范的OrderRef进行报单。

```
virtual void notifyGroupMaxOrderRef(const int groupMaxOrderRef[])
```

对于使用ydExtendedApi的投资者，可以使用下列方法直接获取报单或者报价的下个OrderRef：

```
virtual int getNextOrderRef(unsigned orderGroupID=0,bool update=true)
```

orderGroupID指定报单组编号。若orderGroupID设置为0，可以设置下个OrderRef的规则，则请参考多连接；若orderGroupID设置为1至255，那么下个OrderRef必定为当前最大OrderRef加1。

update指示获取下个OrderRef后，是否需要更新该报单组的最大报单编号。假设本次调用getNextOrderRef返回的下个OrderRef为n，若本次调用中update为true，那么下次调用getNextOrderRef返回的下个OrderRef为n+1；若本次调用中update为false，那么下次调用getNextOrderRef返回的下个OrderRef仍然为n。

7.13 多连接

易达柜台默认支持同一个资金账号同时启动多个API实例连接柜台，而且默认情况下无连接数限制。经纪商可以在柜台端控制每个投资者的登录连接个数上限，投资者可以在YDAccount的MaxLoginCount查询经纪商是否设置了连接上线，当没有限制时MaxLoginCount为-1。同时，YDAccount的LoginCount记录了当前该资金账户连接的总数。对于同一个资金账号的不同连接，MaxLoginCount和LoginCount总是相同的。

若投资者需要区分不同连接的报单，请优先使用报单组功能，**不建议**使用下文提到的原生SessionID机制。

> 在1.386版本中，易达提供了原生的SessionID机制，该机制是为了满足沪深证券交易所的监管要求，设计时没有考虑投资者的交易需求，因此不建议投资者在交易流程中使用本SessionID机制，但是可在日志中记录便于后期排错。该机制在现货柜台默认启用，但在期货柜台默认关闭，若需在期货柜台需要使用，则需联系经纪商在柜台端启用该功能。
>
> SessionID由柜台负责分配，当API初始连接或断线重连后均新分配SessionID。所有投资者的所有连接上限为4096个，SessionID编号为0到4095，系统将随机分配SessionID，不会按照连接先后顺序分配。连接断开后，对应的SessionID将被回收以备重用，因此同一个交易日的不同时间，不同投资者的SessionID在可能重复，同一投资者也可能重复拿到同一SessionID，但是同一时间内各个连接的SessionID必然是不同。因此强烈建议投资者始终在notifyLogin获取getSessionID。获取SessionID的函数如下所示：
>
> ```
> virtual int getSessionID(void)
> ```
>
> 报单和报价的SessionID可在YDOrder.SessionID和YDQuote.SessionID中获得。普通柜台重启或者高可用柜台主备切换后，柜台中发回的报单回报的SessionID会被置为-1。

> 在过去版本的API中，易达为使用ydExtendedApi的投资者提供了一套利用OrderRef的低位编码SessionID的机制，但由于限制较多，易达不再推荐优先使用。为保持兼容性，当前版本中仍然为编号为0的报单组保留该机制，若投资者使用非0报单组则下述内容无效。
>
> 请注意，一旦决定使用该机制，请在所有连接中配置使用该机制，否则会导致潜在的OrderRef编码冲突。对于使用了本地风控报单的多连接投资者，请务必使用SessionID机制，否则所有连接发出的OrderRef都会冲突。
>
> 该连接号机制涉及以下函数：
>
> ```
> virtual bool setSessionOrderRefRule(unsigned sessionBitCount,unsigned sessionID)
> virtual void getSessionOrderRefRule(unsigned *pSessionBitCount,unsigned *pSessionID)
> ```

> 请**在任何报单和报价前**，通过setSessionOrderRefRule设置线程号以及全局线程号规则，setSessionOrderRefRule的各参数说明如下：
>
> - sessionBitCount设置了使用OrderRef中用于SessionID的末尾保留位数，保留位数决定了最大连接数和最大报单数，保留位数最多16位，这个保留位数必须在所有的连接中都相同
> - sessionID则标记了本连接的连接号，请从0开始为每个API实例的连接编号，请确保连接总数不超过sessionBitCount指定的个数范围
>
> 举例来说，假设某连接设置sessionBitCount为8，SessionID为3，那么OrderRef中有8位用于连接号，剩下的24位（32-8=24位）用于真正的报单编号，因此，在这种设置模式下，可以有256（2\^8）个连接号，每个连接可以有16,777,216（2^24）张报单，同时，本API实例的连接号为3。该连接的第一个OrderRef为259（0b100000000+0b11=0b100000011=259）。
>
> 在设置完成后，即可以使用getSessionOrderRefRule获取设置的连接号编码规则，以及使用getNextOrderRef获取下一个报单编号。

7.14 裸协议

若使用`1.280`及以上版本API，请严格按照以下裸协议规范报单和接收回报，否则将会被认为错单。若使用`1.188`及以下版本API，请参考对应版本的裸协议规范。

7.14.1 上行报文

从`1.52`版本开始，易达开放了报单和撤单报文，并在`1.280`版本中增加开放了报价、撤报价以及各种回报报文。投资者可以向易达柜台发送自组的UDP报文进行交易。由于裸协议报单是指用户自己负责数据包的组装和发送，其性能直接取决于用户的实现，和易达API没有关系。无论是通过裸协议报单还是通过易达API报单，在柜台端的穿透性能是没有区别的，因此，与API的性能差异仅仅在于客户端发送端的性能。

易达API经过长时间的迭代，在易达实验室中（在X10/25和X2522网卡上）测得的性能已达到普通FPGA的水准。如果客户尝试自己实现裸协议报单，那么请在实现完成并与易达API的报单性能进行比较后，选择较快的一种作为您生产报单的方式。

API提供了时间戳函数getYDNanoTimestamp，该函数精度高速度快，可以用来测试API的报单速度。具体方法为在insertOrder的前后调用getYDNanoTimestamp并将结果相减，得到的是从开始发送到报单的第一个字节出现在光纤上的时间差。

裸协议报撤单可通过UDP或者XTCP报送，UDP或者XTCP端口号可以在SystemParam中获取，详情参见系统参数。源端口号可以任意，易达不检查源端口号。当使用XTCP裸协议报单时，请每隔2秒向柜台发送心跳报文以维持连接。

7.14.1.1 使用准备

投资者需要得到经纪商的批准后才能使用裸协议，使用前请与经纪商联系开通。投资者可以检查YDAccount.AccountFlag是否设置了YD_AF_BareProtocol以确认是否已经开通了裸协议。

UDP报文头需要通过调用getClientPacketHeader获取，报文头中包含了账户、密钥等信息，因此不可修改报文头中任何内容。只能在notifyFinishInit及其后续步骤中获取报文头。一旦成功获得了报文头，本次柜台启动期间是不变的，如果柜台发生了重启，则需重新获取。现货柜台使用裸协议报单的投资者请注意，由于现货柜台必须打开原生SessionID机制，若盘中发生断线重连，断线前的裸协议报文头将失效，请务必在登录成功后的notifyLogin中重新获取报文头部。

```
/// definition of protocolVersion
///     0: newest protocol version
///     1: protocol for api version up to 1.188
///     2: protocol for api version up to 1.386
///     3: protocol for api version from 1.486, current newest version
virtual int getClientPacketHeader(YDPacketType type,unsigned char *pHeader,int len,int protocolVersion=0)
```

YDPacketType可以是YD_CLIENT_PACKET_INSERT_ORDER、YD_CLIENT_PACKET_CANCEL_ORDER、YD_CLIENT_PACKET_INSERT_QUOTE、YD_CLIENT_PACKET_CANCEL_QUOTE，分别用于获取报单、撤单、报价、撤报价的报文头。1.280版本开始，易达新增了4类特殊的报撤单类型，分别是YD_CLIENT_PACKET_INSERT_NORMAL_ORDER、YD_CLIENT_PACKET_CANCEL_NORMAL_ORDER、YD_CLIENT_PACKET_INSERT_SPECIAL_ORDER和YD_CLIENT_PACKET_CANCEL_SPECIAL_ORDER，前两个只能用于普通报撤单，报入其他业务会报错；后两个只能用于行权、放弃行权、备兑等非交易业务的提交和撤回，报入普通报撤单会报错。

header和len是用户预先创建的存储区域的头指针以及长度，用于容纳api生成的头部数据，len应大于对应类型的头部的长度，目前各业务报文头长度都是16。

protocolVersion可指定协议报文版本，默认为当前最新版本报文版本。选择指定协议报文版本后，需参考相应版本的API文档中的报文格式说明。下表展示了在当前版本中，协议报文版本实际使用的报文格式对应的版本号。对于使用版本号0的投资者需关注当前实现与最新版本中的协议报文是否保持兼容。

版本	报单	撤单	报价	撤报价
1	1.188	1.188	1.280	1.280
2	1.280	1.280	1.280	1.280
3	1.280	1.280	1.486	1.486
0	*1.280*	*1.280*	*1.486*	*1.486*

随着功能增加，新版本的报文长度会增长。出于以下两个原因，投资者需要谨慎选择报文版本：

- 柜台会检查报文头中的报文长度与报文实际长度一致性，当使用不匹配的报文头和报文体时，会被柜台当做错误报文丢弃。例如，升级到1.486版本的API后，该版本默认报价报文头的长度为88，若程序仍按照1.280版本文档发送长度为64的报价单，该报价单会被柜台丢弃。因此，建议投资者明确指定协议报文版本，以免造成升级API版本后的兼容性问题
- 老版本的报文通常更短，客户端组包、发送和柜台端接收报文均具有微小的穿透优势。若不使用新版本的新增功能，穿透敏感型投资者可以选择使用老版本报文，易达承诺新版本柜台兼容老版本报文协议

返回值表示实际返回的报文头长度，如果返回0，表示获取报文头失败，通常是因为预先创建的存储区域长度不够，调用时间点早于notifyFinishInit或者没有开通裸协议报单功能YD_AF_BareProtocol。

7.14.1.2 报单报文

以下为报单时的报文结构，等同于调用insertOrder。

地址偏移	长度(字节)	字段类型	注释
0	16	整数	报文头。
16	4	整数（little-endian）	合约序号。从YDInstrument的InstrumentRef中获取。
20	1	整数	买卖方向(0：买，1：卖)
21	1	整数	开平标志(0：开仓，1：平仓，3：平今，4：平昨)
22	1	整数	投保标志(1：投机，2：套利，3：套保)
23	1	整数	席位选择方法 (0：YD_CS_Any, 1：YD_CS_Fixed, 2：YD_CS_Prefered)
24	8	IEEE 754双精度浮点类型（little-endian）	价格
32	4	整数（little-endian）	报单数量

地址偏移	长度(字节)	字段类型	注释
36	4	整数（little-endian）	报单引用
40	1	整数	报单类型(0：限价单，1：FAK, 2:市价单，3:FOK)
41	1	整数	填0
42	1	整数	席位连接编号
43	5	整数	填0
48	1	无符号整数	报单组ID
49	1	整数	报单组OrderRef控制方式，详情参见报单组 0：OrderRef单调递增，前后两单OrderRef的间隔必须大于等于1 1：OrderRef严格单调递增，前后两单OrderRef的间隔必须为1
50	1	整数	触发单类型 0：无触发条件 1：止盈触发 2：止损触发
51	1	整数	交易所报单属性，由报单所属交易所解释其含义。 YD_EOA_DCE_GIS=1：大商所GIS当前小节有效报单标志
52	4	整数（little-endian）	参考远程用户自定义字段
56	8	整数	填0
64	8	IEEE 754双精度浮点类型（little-endian）	触发单触发价格

7.14.1.3 撤单报文

以下为撤单时的报文结构，等同于调用cancelOrder。目前支持用全精度的交易所报单编号、交易所报单编号、委托编号OrderRef三种撤单方式，具体请参考普通撤单。

地址偏移	长度(字节)	字段类型	注释
0	16	整数	报文头。
16	4	整数（little-endian）	欲撤单的交易所报单编号或者OrderRef。
20	1	整数	交易所序号。从YDExchange的ExchangeRef中获取。
21	1	整数	席位选择方法 (0：YD_CS_Any, 1：YD_CS_Fixed, 2：YD_CS_Prefered)
22	1	整数	席位连接序号
23	1	整数	填0
24	1	无符号整数	报单组ID

地址偏移	长度(字节)	字段类型	注释
25	7	整数	填0
32	8	整数（little-endian）	全精度的交易所报单编号

7.14.1.4 报价报文

以下为报单时的报文结构，等同于调用insertQuote。

地址偏移	长度(字节)	字段类型	注释
0	16	整数	报文头。
16	4	整数（little-endian）	合约序号。从YDInstrument的InstrumentRef中获取。
20	1	整数	买开平标志
21	1	整数	买投保标志
22	1	整数	卖开平标志
23	1	整数	卖投保标志
24	8	IEEE 754双精度浮点类型（little-endian）	买价格
32	8	IEEE 754双精度浮点类型（little-endian）	卖价格
40	4	整数（little-endian）	买量
44	4	整数（little-endian）	卖量
48	4	整数（little-endian）	报单引用，即OrderRef
52	1	整数	席位选择方法 (0：YD_CS_Any, 1：YD_CS_Fixed, 2：YD_CS_Prefered)
53	1	整数	席位连接编号
54	1	整数	填0
55	1	整数	报价标志 YD_YQF_ResponseOfRFQ：自动填写询价号表示应价，支持的有上期所、能源所、大商所、广期所、郑商所，其他交易所无询价号 YD_YQF_ReplaceLastQuote：是否顶最后发送的报价，只对中金所生效 YD_YQF_ReplceSpecifiedQuote：是否顶指定报价，只对中金所生效
56	1	无符号整数	报单组ID
57	1	整数	报单组OrderRef控制方式，详情参见报单组 0：OrderRef单调递增，前后两单OrderRef的间隔必须大于等于1 1：OrderRef严格单调递增，前后两单OrderRef的间隔必须为1
58	1	整数	交易所报价属性，由报价所属交易所解释其含义。 YD_EOA_DCE_GIS=1：大商所GIS当前小节有效报单标志

地址偏移	长度(字节)	字段类型	注释
59	5	整数	填0
64	4	整数（little-endian）	参考远程用户自定义字段
68	12	整数	填0
80	8	整数（little-endian）	中金所顶指定报价时，需要指被顶单的QuoteSysID

7.14.1.5 撤报价报文

以下为撤报价时的报文结构，等同于调用cancelQuote。目前支持用全精度的交易所报单编号、交易所报单编号、委托编号OrderRef三种撤报价方式，具体请参考普通撤报价。

地址偏移	长度(字节)	字段类型	注释
0	16	整数	报文头。
16	4	整数（little-endian）	欲撤报价的交易所报价编号或者OrderRef。
20	1	整数	交易所序号。从YDExchange的ExchangeRef中获取。
21	1	整数	席位选择方法 (0：YD_CS_Any, 1：YD_CS_Fixed, 2：YD_CS_Prefered)
22	1	整数	席位连接序号
23	1	无符号整数	报单组ID
24	8	整数（little-endian）	全精度的交易所报价编号
32	1	整数	撤报价的方式，只对上交所有效，深交所只能两边同时撤： YD_CQIT_Buy=1：撤买边 YD_CQIT_Sell=2：撤卖边 YD_CQIT_Both=3：撤双边
33	7	整数	填0

7.14.2 下行报文

易达的下行报文通过TCP发送，在能高效解决UDP可靠送达的问题之前，易达不会考虑增加UDP下行报文，需要解析下行报文的投资者需要自行旁路监听TCP报文段，并正确处理可能发生的重发、黏连等问题。

7.14.2.1 报文头

易达的下行报文使用相同的头部结构，收到下行报文后，首先需要解析报文头中的报文类型以区分使用何种报文结构解析后续数据。报文头的结构如下表所示：

地址偏移	长度(字节)	字段类型	注释
0	2	整数（little-endian）	报文长度（包含本报文头）
2	2		非公开字段

地址偏移	长度(字节)	字段类型	注释
4	4	整数（little-endian）	报文类型
8	4		非公开字段
12	4	整数（little-endian）	投资者账户序号

上述报文头中可能的报文类型如下表所示：

报文类型	说明
34	报单回报
35	成交回报
37	询价
40	报价回报
43	撤单或撤报价失败回报

7.14.2.2 报单回报报文

由于大量业务复用了报单回报的报文，因此请只关注报单标志为0的报单回报，非0的报单回报应忽略。

地址偏移	长度(字节)	字段类型	注释
0	16		报文头
16	4	整数（little-endian）	合约序号。对应YDInstrument的InstrumentRef。
20	1	整数	买卖方向(0：买，1：卖)
21	1	整数	开平标志(0：开仓，1：平仓，3：平今，4：平昨)
22	1	整数	投保标志(1：投机，2：套利，3：套保)
23	1	整数	席位选择方法 (0：YD_CS_Any, 1：YD_CS_Fixed, 2：YD_CS_Prefered)
24	8	IEEE 754双精度浮点类型（little-endian）	报单价格
32	4	整数（little-endian）	报单数量
36	4	整数（little-endian）	投资者报单编号
40	1	整数	报单类型(0：限价单，1：FAK， 2:市价单，3:FOK)
41	1	整数	报单标志
42	1	整数	指定的席位连接编号
43	1	整数	实际使用的席位连接编号
44	4	整数（little-endian）	错误码
48	4	整数（little-endian）	交易所序号
52	4	整数（little-endian）	交易所报单编号

地址偏移	长度(字节)	字段类型	注释
56	4	整数（little-endian）	报单状态
60	4	整数（little-endian）	成交数量
64	4	整数（little-endian）	交易所报单时间
68	4	整数（little-endian）	柜台本地报单编号
72	1	无符号整数	报单组ID
73	1	整数	报单组OrderRef控制方式，详情参见报单组 0：OrderRef单调递增，前后两单OrderRef的间隔必须大于等于1 1：OrderRef严格单调递增，前后两单OrderRef的间隔必须为1
74	1	整数	触发单类型 0：无触发条件 1：止盈触发 2：止损触发
75	1	整数	交易所报单属性，由报单所属交易所解释其含义。 YD_EOA_DCE_GIS=1：大商所GIS当前小节有效报单标志
76	4	整数（little-endian）	参考远程用户自定义字段
80	8		非公开字段
88	8	IEEE 754双精度浮点类型（little-endian）	触发单触发价格
96	4	整数（little-endian）	报单触发状态 0：未触发 1：已触发
100	4	整数（little-endian）	从该交易日开始（17点整）到报单时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
104	8	整数（little-endian）	全精度的交易所报单编号
112	4	整数（little-endian）	从该交易日开始（17点整）到撤单时间的豪秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
116	4	整数（little-endian）	连接号
120	4		非公开字段
124	4	整数（little-endian）	供国君证券内部使用的报单编号

7.14.2.3 成交回报报文

地址偏移	长度(字节)	字段类型	注释
0	16		报文头
16	4	整数（little-endian）	合约序号。对应YDInstrument的InstrumentRef。
20	1	整数	买卖方向(0：买，1：卖)
21	1	整数	开平标志(0：开仓，1：平仓，3：平今，4：平昨)
22	1	整数	投保标志(1：投机，2：套利，3：套保)
23	1		非公开字段
24	4	整数（little-endian）	交易所成交编号
28	4	整数（little-endian）	交易所报单编号
32	8	IEEE 754双精度浮点类型（little-endian）	成交价格
40	4	整数（little-endian）	成交数量
44	4	整数（little-endian）	交易所成交时间
48	8	IEEE 754双精度浮点类型（little-endian）	成交手续费
56	4	整数（little-endian）	柜台本地报单编号
60	4	整数（little-endian）	投资者报单编号
64	1	无符号整数	报单组ID
65	1	整数	实际使用的席位连接编号
66	2		非公开字段
68	4	整数（little-endian）	从该交易日开始（17点整）到成交时间的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
72	8	整数（little-endian）	全精度的交易所报单编号
80	8	整数（little-endian）	全精度的交易所成交编号
88	4	整数（little-endian）	参考远程用户自定义字段
92	4		非公开字段

7.14.2.4 询价回报报文

地址偏移	长度(字节)	字段类型	注释
0	16		报文头
16	4	整数（little-endian）	合约序号。对应YDInstrument的InstrumentRef。
20	4	整数（little-endian）	从该交易日开始（17点整）到询价时间的秒数。请参考以下例子： 夜盘21点整：3600*(21-17)=14400 日盘9点整：3600*(24+9-17)=57600 周一日盘9点整：3600*(24+9-17)=57600 易达整数时间和标准时间的转换程序参见ydUtil.h中的string2TimeID和timeID2String
24	4	整数（little-endian）	询价ID
28	4		非公开字段
32	8	整数（little-endian）	全精度的询价编号

7.14.2.5 报价回报报文

地址偏移	长度(字节)	字段类型	注释
0	16		报文头
20	4	整数（little-endian）	合约序号。对应YDInstrument的InstrumentRef。
24	1	整数	买开平标志
25	1	整数	买投保标志
26	1	整数	卖开平标志
27	1	整数	卖投保标志
28	8	IEEE 754双精度浮点类型（little-endian）	买价格
36	8	IEEE 754双精度浮点类型（little-endian）	卖价格
44	4	整数（little-endian）	买量
48	4	整数（little-endian）	卖量

地址偏移	长度(字节)	字段类型	注释
52	4	整数（little-endian）	报单客户序号，即OrderRef
56	1	整数	席位选择方法 (0：YD_CS_Any, 1：YD_CS_Fixed, 2：YD_CS_Prefered)
57	1	整数	指定的席位连接编号
58	1	整数	实际使用的席位连接编号
59	1	整数	报价标志 YD_YQF_ResponseOfRFQ：自动填写询价号表示应价，支持的有上期所、能源所、大商所、广期所、郑商所，其他交易所无询价号
60	1	无符号整数	报单组ID
61	1	整数	报单组OrderRef控制方式，详情参见报单组 0：OrderRef单调递增，前后两单OrderRef的间隔必须大于等于1 1：OrderRef严格单调递增，前后两单OrderRef的间隔必须为1
62	1	整数	交易所报价属性，由报价所属交易所解释其含义。 YD_EOA_DCE_GIS=1：大商所GIS当前小节有效报单标志
63	1		非公开字段
64	4	整数（little-endian）	错误号
68	4	整数（little-endian）	交易所序号。对应YDExchange的ExchangeRef。
72	4	整数（little-endian）	当ErrorNo为YD_ERROR_InvalidGroupOrderRef时，表示当前柜台已收到的最大OrderRef；否则表示报价编号，用于撤报价。若交易所返回值超长，则会被截断
76	4	整数（little-endian）	买报价的OrderSysID，卖单边报价时为0。若交易所返回值超长，则会被截断
80	4	整数（little-endian）	卖报价的OrderSysID，买单边报价时为0。若交易所返回值超长，则会被截断
84	4	整数（little-endian）	当YDQuoteFlag为YD_YQF_ResponseOfRFQ时，记录了报单响应的询价号，否则为0。若交易所返回值超长，则会被截断
88	4	整数（little-endian）	连接号
92	8	整数（little-endian）	全精度的报价编号，用于撤报价
100	8	整数（little-endian）	全精度的买报价的OrderSysID，卖单边报价时为0
108	8	整数（little-endian）	全精度的卖报价的OrderSysID，买单边报价时为0

地址偏移	长度(字节)	字段类型	注释
116	8	整数（little-endian）	当YDQuoteFlag为YD_YQF_ResponseOfRFQ时，记录了全精度的报单响应的询价号，否则为0
124	4	整数（little-endian）	参考远程用户自定义字段
128	12		非公开字段
140	8	整数（little-endian）	中金所顶指定报价时被顶单的QuoteSysID

7.14.2.6 撤单或撤报价失败回报报文

由于大量业务复用了撤单或撤报价失败回报报文，因此若报文为撤单失败回报时，请只关注报单标志为0的撤单失败回报，非0的撤单失败回报应忽略。

地址偏移	长度(字节)	字段类型	注释
0	16		报文头
16	4	整数（little-endian）	交易所报单编号或报价编号
20	1	整数（little-endian）	交易所序号。对应YDExchange的ExchangeRef。
21	1	无符号整数	报单组ID
22	1	整数	撤报价的方式，只对上交所有效，深交所只能两边同时撤： YD_CQIT_Buy=1：撤买边 YD_CQIT_Sell=2：撤卖边 YD_CQIT_Both=3：撤双边
23	1	整数	报单标志YDOrderFlag，撤报价失败回报报文无效
24	4	整数（little-endian）	错误码
28	4	整数（little-endian）	报价标志IsQuote。 1：本报文为撤报价失败回报报文 0：本报文为撤单失败回报报文
32	4	整数（little-endian）	投资者报单编号OrderRef
36	4		非公开字段
40	8	整数（little-endian）	全精度的交易所报单编号或报价编号

7.14.3 心跳报文

心跳报文用于维持客户端和服务端之间的TCP连接，客户端和服务端均会发送心跳报文。

地址偏移	长度(字节)	字段类型	注释
0	2	整数（little-endian）	固定为8
2	2		非公开字段

地址偏移	长度(字节)	字段类型	注释
4	4	整数（little-endian）	固定为0

7.15 指定席位

前置是指交易所交易系统中供柜台连接并接收柜台报单和返回回报的服务器，通常会有多个，考虑到每个前置上承载的席位连接数量的差异，以及瞬时业务量的不平衡性，不同前置向交易核心传递报单的性能也会有差异，因此，在交易过程中，投资者应该评估各个前置的性能表现并排除最慢的前置后，在剩余席位上报单。

席位是配置在柜台中的用于连接前置的账户，易达会在合规的前提下努力使得每个席位连接不同的前置，以给投资者最大的选择空间。经纪商在部署易达柜台时，通常会配备与交易所前置数相同的席位，因此，通常情况下易达柜台的席位可以覆盖交易所的所有前置，因此，选择前置与选择席位本质上是相同的。

由于指定席位报单的存在，易达柜台上各个席位的报单量通常是不平衡的，特别是在席位优选结果上报的情况下，容易触发席位流控。建议投资者获取席位信息，了解易达席位流控的机制后，根据生产实际需要，指定席位报单或者指定席位撤单。独占柜台的投资者可以使用席位优选结果上报使得所有账户共享席位优选结果。

7.15.1 获取席位信息

易达柜台支持连接到多个交易所同时交易，每个交易所的席位信息是完全独立的，因此，以下内容均是针对单个交易所而言。

交易所的总席位数可以在YDExchange.ConnectionCount中找到，第一个席位的编号是0，最后一个席位的编号是ConnectionCount-1，易达柜台最大支持64个席位。易达将席位分为公共席位和专属席位两种：

- 公共席位：可以被所有投资者使用的席位。在YDExchange.IsPublicConnectionID[64]列出了所有公共席位，可以通过席位编号查找该席位是否是公共席位，若IsPublicConnectionID[i]是true则说明第i个席位是公共席位，否则就是专属席位。
- 专属席位：只能被指定的投资者使用，每个专属席位可以被指定给多个投资者，每个投资者也可以拥有多个专属席位。在YDAccountExchangeInfo.IsDedicatedConnectionID[64]列出了每个投资者的专属席位信息，若IsDedicatedConnectionID[i]是true则说明第i个席位是该投资者的专属席位，否则就不是。

综上，投资者的**可选席位**是公共席位和专属席位的叠加，**不可选席位**是属于其他投资者的专属席位。

投资者会有同前置比较两个柜台性能的需求。在1.386版本中，投资者可以通过YDExchange.ConnectionInfos获取连接信息（可直接通过数组下标访问，数组大小为YDExchange.ConnectionCount），同时，易达API也通过notifyExchangeConnectionInfo在notifyFinishedInit之后告知各连接信息，首次启动时告知所有连接状态，盘中若有席位因断线等原因导致信息变化，亦会通过上述回调通知。

```
1 virtual void notifyExchangeConnectionInfo(const YDExchangeConnectionInfo
  *pExchangeConnectionInfo)
```

其中，YDExchangeConnectionInfo的字段信息如下：

字段	说明
ExchangeRef	交易所序号。可在YDExchange获取。
ConnectionID	连接编号
ConnectionStatus	连接状态 YD_ECS_DISCONNECTED=0：席位断开连接 YD_ECS_CONNECTED=1：席位已连接
Info	前置IP地址信息。默认不发布，如需发布请联系经纪商

字段	说明
InsertFlowControl	在报单或者报价时检查时间窗口内每接口报单数量上限。表达形式为count/window，其中分子count表示数量上限，分母window表示检查时间窗口大小。分子小于等于0表示无上限，不指定分母或者分母小于等于1表示1000。 例如50/1000表示在1000毫秒的滑动时间窗口中不允许报单超过50单。 通常中金所为50/1000，其他交易所为100/1000
CancelFlowControl	在撤单或者撤报价时检查时间窗口内每接口撤单数量上限。表达形式为count/window，其中分子count表示数量上限，分母window表示检查时间窗口大小。分子小于等于0表示无上限，不指定分母或者分母小于等于1表示1000，本字段可留空，表示与席位报单速度上限相同。 例如50/1000表示在1000毫秒的滑动时间窗口中不允许撤单超过50。 通常中金所为50/1000，其他交易所为100/1000

7.15.2 席位流控

目前各个交易所存在两种限制席位发单速度的席位流控模式，请注意席位流控是不区分投资者的，柜台上所有投资者在同一席位上的报单是汇总计算的：

- 滑动窗口流控：限制每个席位或者网关在以1秒（滑动窗口长度可设置）为长度的滑动窗口内的最大报单数量，若超过限制条件，各交易所API的具体行为是不同，有的交易所API会直接拒单，有的则会缓存报单等到下一秒发送。目前所有交易所都有此席位流控模式，且都公布了流控阈值，具体阈值信息请参见下表
- 在途单流控：限制了已经发送到交易所但是还没有收到回报的报单数量，若超过限制条件，各交易所API的具体行为不同，有的交易所API会直接拒单，有的交易所API会排队直到在途单流控低于阈值。目前只有部分交易所有在途单流控，包括上交所和深交所的流速权，但其他期货交易所均没有公布在途单流控的具体规则和阈值。

交易所	滑动窗口流控阈值	在途单流控阈值
中金所	50	
上期所、能源所、大商所、广期所、郑商所	100	
上交所、深交所		流速权

为了避免报单因为滑动窗口流控和在途单流控而导致报单被缓存和延误，以及满足尽可能提前报告错误的原则，易达柜台实现了对滑动窗口流控和在途单流控（从1.386.40.24版本开始，支持上期所和中金所的在途单流控，其他交易所均不支持。由于目前上交所和深交所报单量较小，暂时使用滑动窗口流控替代，如果实际需要请与我们联系），在报单进入API之前拦截会导致流控的报撤单，具体的报错信息请参考指定席位报单。本文所指流控均为滑动窗口流控和在途单流控的统称。

7.15.3 指定席位报单

投资者可以设置YDInputOrder和YDInputQuote中的ConnectionSelectionType和ConnectionID以不同的方式选择报单席位。各种类型的席位优选方式的选择逻辑如下所示，请注意，断线、达到流控和不可选的席位不会参与席位选择。

- ConnectionSelectionType为YD_CS_Any时，会实现全局轮询的效果，具体规则为：
  - 会从上一次使用YD_CS_Any报单的席位的下一个席位开始逐一遍历所有席位（遍历遇到最后一个席位后会从头开始，直到遇到遍历的起始席位为止），若找到某个席位没有排队，则直接选中该席位，否则，判断该席位的报单队列长度是否小于之前的所有席位，若是最小的则记录该席位，直到遍历结束，选中记录的队列长度的最小的席位。最后把报单插入选中席位队列中排队等待报送。
  - 轮询并不区分投资者，所有投资者的报单是统一参与轮询的，因此，某投资者使用本模式报单可能并不能从RealConnectionID中获得连续的席位号。
- ConnectionSelectionType为YD_CS_Fixed时，会实现指定席位报单的效果，具体规则为：
  - 直接选中ConnectionID指定席位并将报单插入队列排队等待报送

- ConnectionSelectionType为YD_CS_Prefered时，会实现首选选择指定席位报单，遇忙使用其他席位的效果，具体规则为：
    - 会从ConnectionID指定席位开始逐一遍历所有席位（遍历遇到最后一个席位后会从头开始，直到遇到遍历的起始席位为止），若找到某个席位没有排队，则直接选中该席位，否则，判断该席位的报单队列长度是否小于之前的所有席位，若是最小的则记录该席位，直到遍历结束，选中记录的队列长度的最小的席位。最后把报单插入选中席位队列中排队等待报送。

在某些情况下，可能会导致没有席位被选中的情况，易达以不同的错误代码区分了在寻找选中席位时可能存在的多种错误情况：

- 一个或多个席位遇到滑动窗口流控或者在途单流控，且未达到流控的席位都处于不可用状态时（属于他人的专属席位、席位断线、到达席位队列上限），则报"YD_ERROR_CanNotSendToExchangeForFlowControl=79没有可用席位，且有席位被流控"的错误
- 如果所有席位都处于不可用状态时（属于他人的专属席位、席位断线、到达席位队列上限），则报"YD_ERROR_CanNotSendToExchange=9没有可用席位"的错误

当已经在席位队列中排队的报单移动到席位队列的队首，并开始向交易所发送时，如果报单、报价、组合、行权等交易所API因为在途单超过上限或者席位断线返回错误的，则报"YD_ERROR_ExchangeConnectionSendError=80交易所API发送错误"。当发生上述错误且投资者的YDAccountFlag打开了柜台回报YD_AF_NotifyOrderAccept的功能时，收到的notifyOrder回报会随机遇到以下两种情况中的一种，所以投资者的策略程序不应该依赖于回报的个数，而是应该关注回报的状态：

- 收到一个OrderStatus=YD_OS_Rejected和ErrorNo=YD_ERROR_ExchangeConnectionSendError的回报
- 收到两个回报，其中第一个是OrderStatus=YD_OS_Accepted和ErrorNo=0的回报，第二个是OrderStatus=Rejected和ErrorNo=YD_ERROR_ExchangeConnectionSendError的回报

被选中的席位（报单实际发送使用的席位）可通过回报YDOrder或者YDQuote中的RealConnectionID获得。

7.15.4 指定席位撤单

投资者可以设置YDCancelOrder和YDCancelQuote中的ConnectionSelectionType和ConnectionID以不同的方式选择撤单席位，与报单不同，撤单时会受到柜台席位类型的限制而不能完全按照投资者设置进行撤单席位选择。

- 当柜台席位类型为全管理席位时，处理逻辑同报单指定席位的逻辑
- 当柜台席位类型为首席位管理其他席位非管理时，则按序检查原报单席位和首管理席位，若找到某个席位没有排队，则直接选中该席位，否则，选中队列长度较小的席位，若队列长度相同则选中原报单席位。最后把报单插入选中席位队列中排队等待报送。
- 当柜台席位类型为全非管理席位时，撤单必须从原报单席位发出，此时柜台直接选中原报单席位并将报单插入席位队列排队等待报送

与指定席位报单一样，指定席位撤单会以完全相同的原因导致错误，详情请参见指定席位报单。

API中没有渠道能获取在撤单中被选中的席位，如排查问题时需要获取该信息的，请联系经纪商在柜台的inputFlow.txt中协助查询。

7.15.5 席位优选结果上报

为了避开较慢的交易所前置，投资者会尝试评估各个前置的性能，将较慢的交易所前置排除后，通过YD_CS_Fixed或者YD_CS_Prefered指定较快的席位报单（前置和席位的对照关系参考获取席位信息）。具体评估前置性能的方法需要由投资者根据策略的具体需求自行实现，柜台不提供任何优选席位的功能，席位优选的原理如下：首先以极小间隔向所有席位发YD_CS_Fixed报单，各个席位回报中交易所报单编号OrderSysID从小到大排列后即得到本轮探测结果，然后通过多轮次的探测以获得众多的探测轮次结果，最后统计分析探测结果面板数据，排除统计上较慢的席位。评估不要过于频繁，否则可能会导致柜台堵单，扰乱柜台正常交易，甚至会被交易所认定为异常交易行为。

如果经纪商具备优选能力并希望其客户能使用其优选结果，或者独占柜台中使用其中一个账户进行优选并希望其他账户也能按照优选结果进行报单的，可以使用优选席位上报接口selectConnections定期向柜台传输席位优选结果，分享给同一柜台上的其他账户使用。只有YDAccount.AccountFlag设置了YD_AF_SelectConnection标志的投资者以及管理员才能调用该接口。

```
virtual bool selectConnections(const YDExchange *pExchange,unsigned long long connectionList)
```

connectionList可以看做是一个元素长度为4位序列，每4位表示一个席位编号，最低的位表示最快的席位，必须指定交换的所有席位编号。例如，要设置由快到慢的席位（前置）的顺序为2-3-0-1-4，那么connectionList的二进制位表示为：0100 0001 0000 0011 0010。

优选席位结果的有效期为MaxSelectConnectionGap（默认为5分钟），超时后上报结果过期失效，因此应在MaxSelectConnectionGap时限内再次提交新的上报结果，以防止席位优选机制过期失效；同时，席位优选结果上报间隔也不宜过于频繁，最小间隔不可以低于MinSelectConnectionGap（默认60秒）。MaxSelectConnectionGap和MinSelectConnectionGap的获取方式请参见系统参数。

当柜台上的席位优选结果生效时，该柜台上所有使用YD_CS_Any方式的报撤单都将按照席位优选结果的优先级选择席位发单，席位选择规则为：按照席位优选结果的顺序逐一遍历所有席位，优先选择没有排队的席位，若所有席位都有排队，则选择队列长度最短的席位。选择过程中，不可用的席位将被跳过，如席位流控、属于他人的专属席位、席位断线、到达席位队列上限等。

在席位优选结果生效的柜台上，由于报单倾向于按照优选顺序选择较快的席位，会导致较快的席位更容易发生流控，当柜台配置为全非管理席位时，会造成被流控的席位上的挂单无法撤单的情况。同时，为了规避同前置压制效应，从1.280版本开始易达柜台支持设置优选席位轮转个数，该功能当且仅当柜台席位优选结果生效时起效。

> 同前置压制效应是指在生产中，两套相同品牌和配置的柜台先后对同一个前置发送报单时，先发单的柜台始终领先后发单的柜台的情况。发生此类问题的原因是，两套柜台逻辑上报送到同前置的报单存在先后关系，即便只差极其微小的时间差（举例而言100ns），也会存在绝对的先后顺序关系，从而导致交易所优选处理先到的报单。但事实上这两套柜台的性能无本质差异，不应造成如此悬殊的结果。此时，如果使用优选排名第二或第三的席位报单，可能会把领先概率板回，因为本质上席位优选只是排除较差的席位，而排在前列的席位的性能可能实质上是接近的，因此，避开使用相同前置报单可能可以缓解同前置压制问题。

设置席位轮转个数n后，每次报单时将循环使用前n个优选席位报单。假设优选结果仍为2-3-0-1-4-5，席位轮转个数设置为4，表示轮转前4个席位，即2、3、0、1。席位选择规则与没有设置席位轮转个数的时候基本一致，唯一区别在于设置席位轮转个数后，每次报单的"实际生效"席位优选结果会按照席位轮转改变：

- 第1张报单选择席位时参考的席位优选结果顺序为2-3-0-1-4-5
- 第2张报单选择席位时参考的席位优选结果顺序为3-0-1-2-4-5
- 第3张报单选择席位时参考的席位优选结果顺序为0-1-2-3-4-5
- 第4张报单选择席位时参考的席位优选结果顺序为1-2-3-0-4-5
- 第5张报单选择席位时参考的席位优选结果顺序为2-3-0-1-4-5
- 以此类推

7.16 超时未知单处理

超时未知单是指柜台向交易所发单超过一定时间后没有收到交易所回报的报单，因为没有交易所回报，所以报单的状态不会发生改变，该报单会停留在柜台中持续冻结保证金或者持仓而无法释放。超时未知单通常发生在与交易所断线时候报单或者交易所切换状态时，但是发生概率极低。易达提供了处理超时未知单的功能，但是在使用时请严格按照以下说明进行，在确认是超时未知单的基础上再进行处理，以免造成损失。

易达柜台会持续监测处于YD_OS_Accepted的报单（询价单和大商所、广期所YDOrderFlag为YD_YOF_Mark的指令除外），若该报单已经超过MissingOrderGap（默认值为60秒，经纪商可以修改，详情参见系统参数）没有收到柜台回报，那么就会通过notifyMissingOrder告知投资者有一个指超时未知单：

```
virtual void notifyMissingOrder(const YDMissingOrder *pMissingOrder)
```

超时未知单结构YDMissingOrder与YDOrder是相同的，但是在返回时YDMissingOrder.InsertTime填写的是柜台收到的时间，这点与YDOrder.InsertTime是交易所时间是有区别的，请注意区分。

投资者在收到通知后，请先在主席柜台确认该报单的状态：

- 如果主席上有该报单的信息，那么很显然该报单已经被交易所接收且交易所给出的回报已经丢失
- 如果主席上没有该报单的信息，则说明要么该回报没有发送到交易所，要么交易所还没有发送回报，考虑到生产中交易所曾经发生过回报超过1分钟还没有回来的情况，所以建议再等待一段时间

如果已经确认该报单是超时未知单，那么请联系经纪商使其帮助处理，经纪商业务人员也应该遵守上述规则经过**审慎**判断后才能帮助投资者撤销该超时未知单。超时未知单只能由经纪商的管理员账户撤销，投资者自身不可撤销。在经纪商执行了超时未知单撤单操作后，该超时未知单的撤单回报会通过notifyOrder返回，返回的YDOrder.ErrorNo为YD_ERROR_InternalRejected，YDOrder.OrderStatus为YD_OS_Rejected。

如果在撤回了"超时未知单"后收到了交易所的报单回报，易达柜台会将其当做外部报单处理，而外部报单的OrderRef和OrderLocalID都为-1，这样就失去了和原报单关联的机会，当然此时投资者是可以撤回该报单的。如果后续继续收到了成交回报，易达会更新持仓并向投资者发送成交回报，因此投资者不用担心持仓和资金的正确性。

7.17 性能调优

性能调优的目标是降低总体穿透，即从接收到行情开始，到报单请求从柜台发出到交易所，包含了行情接收、策略计算、客户端发单、交换机转发、柜台风控并发送到交易所等步骤。上述的每个步骤都在交易的关键路径上，一个步骤的低性能将会抵消其他步骤在高性能方面的努力，只有每个步骤都做到最高穿透性能，才能保证领先于市场。

其中，客户端发单和柜台风控并发送到交易所是易达提供的交易服务的关键步骤，这两个步骤的穿透时间分别称为API穿透性能和柜台穿透性能。

7.17.1 API穿透性能

API穿透性能是指开始调用insertOrder(或者是其他报单方法)到报单的第一个字节出现在光纤上的时间差。

由于大部分柜台的API在调用后即刻返回，想要测试API穿透性能的唯一办法是将投资者策略机上行柜台的TX口分光后回接到策略机，并保持回接网卡时钟与系统时钟同步。在报单前记录下系统时钟的时间戳作为开始时间戳，使用tcpdump或者类似工具给回接网卡收到的报单包打上时间戳作为结束时间戳，两个时间戳相减即为API的穿透性能。回接网卡需支持纳秒精度硬件时间戳，且使用对应的工具与系统时间保持同步，若使用tcpdump工具，可参考使用如下命令：

```
tcpdump -i <if_name> -n --time-stamp-type=adapter_unsynced --time-stamp-precision=nano -w
<pcap_file>
```

易达API的发单函数返回就表示已将数据包发送到了光纤上，因此，可以直接在报单前后记录系统时间戳并计算差值的简易方法取得穿透性能。建议投资者按照前述API穿透性能标准测试方法测量易达API的穿透性能，并与简易方法相比较，若最终结论为标准测量方法和简易测量方法的结果基本一致，则后续即可使用简易方法测量易达API。请注意，在使用简易方法测量其他API或者裸协议报单前，请务必先使用标注方法校准简易方法的可行性。

易达提供了快速且高精度的时间戳函数，性能开销小，不会对报单关键路径产生过多的影响，在测试API穿透性能时，通过在报单函数前后使用该函数获得时间戳，相减后即得到了API穿透的纳秒数。

```
// Returns nanoseconds elapsed since current process starts up
YD_API_EXPORT unsigned long long getYDNanoTimestamp();
```

易达API经过长时间的迭代优化，其在使用Solarflare X2522或者Exanic X10/X25网卡并配合其加速软件的情况下，API穿透性能达到250ns以下，若投资者实际测得的穿透性能未达到这个指标的，请首先检查网卡、启动方式是否符合易达要求并努力达到易达官方的性能标准。如仍无法达到性能标准，请使用ydcts工具（打开CTS.MeasureAPI=yes功能）并观察API穿透性能，若ydcts可以达到性能标准，请检查策略程序是否存在改进空间，若ydcts仍然无法到达性能标准，请通过经纪商与易达取得联系。

曾经在其他系统上使用过裸协议的投资者会倾向于使用裸协议，易达为了照顾投资者的交易习惯也提供了裸协议报撤单的方法。但易达API的穿透性能已经达到了普通FPGA的水准，因此，如果投资者尝试自己实现裸协议报单，那么请在实现完成裸协议报单后与易达API的穿透性能进行比较，最终选择较快的一种作为您生产报单的方式。

7.17.2 柜台穿透性能

柜台穿透性能指报单第一个字节进入和离开柜台的时间差，该指标可客观公正的衡量柜台的性能，是最常用的判断柜台性能的指标。

业内通常使用分光器或者端口镜像的方式进行测量。分光器精度高成本低但不易调整，适合于临时搭建测量；端口镜像易于调整，但只有Arista 7130等高端型号提供的端口镜像镜像功能可同时做到对最小性能影响和最高测量精度，其他中低端型号并不适合用来测量柜台穿透。易达向经纪商提供了专门的工具用来测量和分析柜台上每笔报单、报价和撤单的上行穿透性能，如需了解您的报单穿透数据可向经纪商索取。

绝大多数情况下，穿透性能不应该是投资者应该关注的问题，易达对每个版本都做了全方位的测试，确保在生产环境中可以达到易达预期的性能指标。当投资者怀疑因穿透性能恶化而导致收益下降时，可与经纪商联系查看穿透是否符合性能标准。标准性能会随每台服务器测试完成时提供给经纪商。

在某些特殊的交易行为情况下，柜台的穿透性能会恶化。目前已知柜台上的报单发送速度太快或者太慢都有可能会显著增加穿透延时，请分别予以应对。若不是这些交易行为引起的穿透性能恶化，请及时联系易达客服排查解决。

7.17.2.1 堵单

堵单发生于多个报单以低于穿透延时的间隔到达柜台，且指定从同一个席位发出。由于交易所是TCP连接，报单需要排队发出，一方面同时到达，一方面排队发出，这就导致了从穿透性能上会体现为，除了第一张单子穿透延时正常外，接下来每张报单的穿透延都会相对上一张报单增加一个较为固定的延时，这样就形成了堵单。

发生堵单可能是以下情况的一种或者多种情况的叠加。

- 报单都选择同一个席位用Fix排队发出（Any和Preferred则不会排队）。特别容易出现在做了席位优选的条件下，不同的投资者会不约而同持续使用其优选的最佳席位报单。
- 有投资者过于频繁的报单，导致席位处于忙的总体时间增加，造成后续Fix指令有更大的几率排队。例如有客户在以较高频率发送席位优选单且没有避开行情切片的到达时间。
- 行情切片到来或者交易节开始起到了发令枪的效果，大部分投资者都会在同一时间点抢发单。
- 有投资者使用了错误的预热方法导致柜台收到巨大的报单量。例如，以极高的频率发送撤单到柜台以期望获取预热效果。
- 同一时间向不同的席位发单也会导致延时的依次增加，但增加量相对同一席位的情况缓解很多，比较常见的情况是在对所有席位发优选单时，后面席位的优选单穿透会少许增加。

堵单本质上是正常的交易行为引起的，易达除了进一步降低穿透外，没有更好的办法来缓解这类问题。对于经纪商，如果发现有投资者采取的错误的预热方式可以及时劝阻，有投资者席位优选单频率过高或者没有避开行情时建议他调整优选单的发送时机，将互相冲突的严重的投资者分开在不同的柜台上。

7.17.2.2 慢速报单

柜台上的报单速度太慢时会导致柜台服务器的缓存变冷，新的报单来的时候该报单的穿透会显著增加。绝大多数时候柜台不会遇到这个问题，因为在柜台上交易的所有投资者的报单都会使缓存变热，只有柜台上所有投资者都以极低的速度（例如整个柜台1单/分钟的频率）报单时，才可能出现这个问题。当投资者与经纪商确认确实遇到这种情况时，建议投资者提前发送预热单以规避这个问题。下面将说明预热单的发送方式和发送时机。

预热单和正式单走相同的执行路径能获得最佳预热效果。

- 预热单和正式单类型相同时执行路径相同，预热效果最好。例如，正式单是报单时，用报单预热的效果优于撤单的。
- 预热单的预热效果仅限于预热单指定的席位，对其他席位的预热效果有限。
- 没能到达交易所的预热单的执行路径短于正式单，预热效果不能达到最佳。
- 预热单和正式单从同一个席位出去但是使用的不同的合约，此时执行路径是相同的，可以达到预热效果。

预热单和正式报单的间隔也会影响预热效果，测试表明两者间隔5秒发送就可以获得最佳的预热效果，更短的间隔并不能获得更好的效果，反而会在收到回报前冻结更长时间的报单保证金，也会增加柜台的压力甚至堵单。同时，预热单要避开行情，否则可能堵住正常的单子，由于预热单并不密集，所以做到这一点相对容易。

不同投资者的预热效果互不影响，不用担心其他客户的预热行为会影响到您的预热效果，反而对您的正式报单也有一定的预热效果。

根据上述预热效果的发送条件，即预热单要做到和正式单类型和席位相同，计算出行情到来的时间提前发送预热单，确保其发送到交易所，这种精确预热是最为推荐的做法。对于在定时发送席位优选单的投资者，席位优选单也是一种可行的预热单，但效果不如精确预热。

8 行情

8.1 前置行情

易达柜台从交易所前置获取一档行情并转发给投资者，易达提供行情主要用于持仓盈亏、保证金的刷新计算，以及在组播行情中断的情况下提供备用行情使用。由于前置一档行情慢于交易所组播行情，为避免错过交易机会，不建议在生产环境中使用易达转发行情作为主力行情。

对于RecalcMode设置为auto或者subscribeOnly的投资者，获取**持仓合约行情**最简单的方法是通过合约指针中的行情指针m_pMarketData直接查询对应合约的行情，详情请参见资金刷新机制章节内容。

对于需要获取非持仓合约行情，或者RecalcMode为off，或者需要在行情更新时收到通知的投资者，可以通过自行主动订阅的方式实现。主动订阅行情前，需要确保API参数按照如下方式配置（RecalcMode设置为auto或者subscribeOnly时API会覆盖以下参数，所以RecalcMode设置为auto或者subscribeOnly时可以不用关心API行情参数的设置）。因为易达柜台不建议使用UDP行情，因此ReceiveUDPMarketData必须始终保持为no，否则将收不到任何行情。

```
ConnectTCPMarketData=yes
ReceiveUDPMarketData=no
```

当ConnectTCPMarketData设置为yes时，启动API后会创建一个单独的行情线程，为了避免行情线程影响到策略系统中其他线程的运行，可以配置API参数将行情线程绑定到某个CPU上面。

```
# Affinity CPU ID for thread to receive TCP market data, -1 indicate no need to set CPU affinity
TCPMarketDataCPUID=-1
```

如果希望行情尽可能快得通知到策略程序，可以设置行情接收线程的工作方式：

- 当设置为-1时系统处于忙查询状态，行情通知性能最佳，但是线程将占满运行所在CPU核心
- 如果不希望行情线程占用过多CPU，且对于行情通知性能并不特别在意的投资者，可以设置select函数的超时时间，这种模式相比忙查询较慢，但是CPU占用将大大降低。推荐投资者在GUI客户端中设置超时时间，以减少不必要的性能开销。

```
# Timeout of select() when receiving TCP market data, in millisec. -1 indicates running without select()
TCPMarketDataTimeout=10
```

关于上述参数的详细说明可以参见行情配置。

以下API提供了订阅和退订某一合约行情的功能，凡是主动订阅了的合约行情，都可以通过上述合约指针中的行情指针m_pMarketData直接查询。

```
virtual bool subscribe(const YDInstrument *pInstrument)
virtual bool unsubscribe(const YDInstrument *pInstrument)
```

合约行情更新后，会通过以下回调函数通知到投资者。

```
virtual void notifyMarketData(const YDMarketData *pMarketData) {}
```

notifyMarketData的返回的参数说明如下：

参数	字段	说明
YDMarketData	InstrumentRef	合约序号

参数	字段	说明
	TradingDay	交易日
	PreSettlementPrice	昨结算价
	PreClosePrice	昨收盘价
	PreOpenInterest	单边昨持仓量
	UpperLimitPrice	涨停板价
	LowerLimitPrice	跌停板价
	LastPrice	最新价
	BidPrice	买一价
	AskPrice	卖一价
	BidVolume	买一量
	AskVolume	卖一量
	Turnover	名义成交金额
	OpenInterest	单边持仓量
	Volume	当日成交量
	TimeStamp	从该交易日开始（17点整）到行情切片的毫秒数。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	AveragePrice	成交均价
	DynamicBasePrice	动态基准价。详情请参考价格偏离度风控。
	LastTradeTimeStamp	从该交易日开始（17点整）到近一次发生交易的毫秒数，仅用于上交所和深交所。请参考以下例子： 夜盘21点过500毫秒：3600*(21-17)*1000+500=14400500 日盘9点500毫秒：3600*(24+9-17)*1000+500=57600500 周一日盘9点500毫秒：3600*(24+9-17)*1000+500=5760500 易达时间戳时间和标准时间的转换程序参见ydUtil.h中的string2TimeStamp和timeStamp2String
	m_pInstrument	指向合约的指针

8.2 组播行情

为满足期货公司内部非展示用途使用组播行情的需求，易达提供了组播行情服务，仅支持上期所和能源所第二代行情，不会支持其他交易所的组播行情。

为方便期货公司从直连交易所迁移到易达行情系统，本系统的使用方式和接口基本上与交易所接口保持一致，详情请参考《上期所第二代行情发布平台接口规范》和《上海期货交易所第二代行情编解码示例》。

为了简化使用场景，行情系统与交易所行情系统的差异如下所示，请期货公司用户注意：

- 本系统能正常处理登录请求，但不会验证用户名口令，任何登录请求都给出成功响应

- 本系统支持同时转发两家交易所的行情。用户需为每个要接收的交易所建立一个连接。本系统内部不区分连接，任何一个连接都可以查询任何一家交易所的信息
- 本系统支持五档或一档行情服务，实际提供的行情服务由系统后台配置决定
- 本系统支持查询最新快照，但是不支持查询历史快照服务
- 本系统**不支持**增量行情查询，如果发送这类请求，会被直接断线

9 风控

易达的风控规则分为事前风控和事后风控两类。

事前风控会拦截不合规的报单，使其不能到达交易所，事前风控通常是监管要求的风控，一旦违反可能会引起禁止交易等监管处罚，如自成交、撤单数限制、仓位限制等。事前风控由柜台在向交易所发送指令前执行，若不通过将会向API返回错单回报。使用ydExtendedApi的投资者可以使用checkAndInsert系列方法在API本地执行风控检查，若通过则正常向柜台发单，若失败则函数返回False，投资者可以在YDInputOrder或者YDInputQuote内查看ErrorNo。需要注意的是，即便API本地执行了风控检查，柜台在处理报单的过程中仍然会再次执行风控检查。

事后风控是经纪商或者投资者自身关注的风控，违反这些风控规则会增加经纪商和投资者的风险，因为对风控阈值的边界是较为宽松的，即便稍微突破一点也不会有质的变化，如信息量等风控规则。但是，若业务上需要保证极端条件下（例如，投资者以极短间隔报单，当事后风控收到回报时已经超过阈值较多）阈值的效力，可以设置一个更强的风控阈值以避免极端条件下超过阈值太多。例如，业务上持仓量风控的阈值要求是100手，那么可以设置为90，留出了10手的超出空间。事后风控一般的处置方式是自动调整投资者的交易权限或者发送风控报警。

9.1 事前风控

9.1.1 事前开仓量风控

开仓量风控分为产品层和合约层两个维度，每个维度上可以分别对开仓量、买开仓量和卖开仓量进行控制。下面以开仓量为例进行说明，买开仓量和卖开仓量风控的处理方法是类似的。

产品层的风控规则是对属于该产品的所有合约的开仓量加总进行汇总风控，一旦触发了产品层风控阈值，该产品内所有合约都无法继续发送开仓指令。

合约层的风控规则是针对单个合约的开仓量加总后进行风控，一旦触发了合约层风控阈值，该合约无法继续发送开仓指令。

本风控并非在开仓量到达阈值以后才开始风控，否则新的开仓报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能开仓量，当开仓报单报出去以后易达就会增加可能开仓量，处于挂单状态时不改变可能开仓量，报单状态达到终态后从可能开仓量中扣减未成交开仓量。因此，如果当前有大量开仓挂单也会导致后续开仓报单因为本风控而被拒绝。

产品层的各个风控参数可以参考YDAccountProductInfo.TradingConstraints[HedgeFlag]的OpenLimit以及DirectionOpenLimit[2]，合约层的各个风控参数可以参考YDAccountInstrumentInfo.TradingConstraints[HedgeFlag]的OpenLimit以及DirectionOpenLimit[2]，具体字段含义请参考风控参数。

9.1.2 事前成交量风控

成交量风控分为产品层和合约层两个维度。

产品层的风控规则是对属于该产品的所有合约的买卖开平成交量加总进行汇总风控，一旦触发了产品层风控阈值，该产品内所有合约都无法继续发送任何指令。

合约层的风控规则是针对单个合约的买卖开平成交量加总后进行风控，一旦触发了合约层风控阈值，该合约无法继续发送任何指令。

本风控并非在成交量到达阈值以后才开始风控，否则新的报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能成交量，当报单报出去以后易达就会增加可能成交量，处于挂单状态不改变可能成交量，报单状态达到终态后从可能成交量中扣减未成交报单量。因此，如果当前有大量挂单也会导致后续报单因为本风控而被拒绝。

产品层的各个风控参数可以参考YDAccountProductInfo.TradingConstraints[HedgeFlag]的TradeVolumeLimit，合约层的各个风控参数可以参考YDAccountInstrumentInfo.TradingConstraints[HedgeFlag]的TradeVolumeLimit，具体字段含义请参考风控参数。

9.1.3 事前持仓量风控

持仓量风控分为产品层和合约层两个维度，每个维度上可以分别对持仓量、买持仓量和卖持仓量进行控制。下面以持仓量为例进行说明，买持仓量和卖持仓量风控的处理方法是类似的。

> **Note**
>
> 从1.486.96.57开始，易达根据交易所规则修订了期权多空持仓的汇总规则，若需使用期权的事前持仓量风控，则需确保柜台版本升级至上述版本或以上版本。
>
> 按照各交易所现行的限仓制度，“期权限仓是指交易所规定非期货公司会员、境外特殊非经纪参与者和客户可以持有的，按单边计算的某月份期权合约投机持仓的最大数量”，但由于事前持仓风控不支持按照系列的汇总，因此易达事前持仓量风控的统计口径与交易所有所不同，不可以直接套用。

产品层的风控规则是对属于该产品的所有合约的买卖持仓量加总进行汇总风控，一旦触发了产品层风控阈值，该产品内所有合约都无法继续发送开仓指令。各指标的汇总规则如下：

- 持仓量：对于期货合约，属于同一产品的期货合约所有持仓的加总；对于期权合约，属于同一产品的期权合约所有持仓的加总
- 买持仓量：对于期货合约，属于同一产品的期货合约买持仓的加总；对于期权合约，属于同一产品的看涨期权的买持仓量和看跌期权的卖持仓量之和
- 卖持仓量：对于期货合约，属于同一产品的期货合约卖持仓的加总；对于期权合约，属于同一产品的看跌期权的买持仓量和看涨期权的卖持仓量之和

合约层的风控规则是针对单个合约的买卖持仓量加总后进行风控，一旦触发了合约层风控阈值，该合约无法继续发送开仓指令。

本风控并非在持仓量到达阈值以后才开始风控，否则新的开仓报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能持仓量，当开仓报单报出去以后易达就会增加可能持仓量，处于挂单状态时不改变可能持仓量，报单状态达到终态后从可能持仓量中扣减未成交报单量。因此，如果当前有大量开仓挂单也会导致后续开仓报单因为本风控而被拒绝。

产品层的各个风控参数可以参考YDAccountProductInfo.TradingConstraints[HedgeFlag]的PositionLimit以及DirectionPositionLimit[2]，合约层的各个风控参数可以参考YDAccountInstrumentInfo.TradingConstraints[HedgeFlag]的PositionLimit以及DirectionPositionLimit[2]，具体字段含义请参考风控参数。

9.1.4 现货买入量风控

现货买入量风控可对投资者的现货日买入量进行控制，控制对象为单个现货证券的日买入量，一旦触发了风控阈值，该现货证券无法继续发送买入指令。

本风控并非在买入量到达阈值以后才开始风控，否则新的买入报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能买入量，当买入报单报出去以后易达就会增加可能买入量，处于挂单状态时不改变可能买入量，报单状态达到终态后从可能买入量中扣减未成交买入量。因此，如果当前有大量买入挂单也会导致后续买入报单因为本风控而被拒绝。

现货买入量风控参数如下表所示：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_ExchangeBuyVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDExchange.ExchangeRef，唯一确定交易所
	IntValue1	最大日买入量
产品级	GeneralRiskParamType	固定为YD_GRPT_ProductBuyVolume

层级	字段	说明
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDProduct.ProductRef，唯一确定产品
	IntValue1	最大日买入量
合约级	GeneralRiskParamType	固定为YD_GRPT_InstrumentBuyVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDInstrument.InstrumentRef，唯一确定合约
	IntValue1	最大日买入量

由于各个层级中AccountRef可以为全体账户或者指定账户两种，因此两两组合一共六个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 合约级且指定账户
- 产品级且指定账户
- 交易所级且指定账户
- 合约所级且对所有投资者
- 产品所级且对所有投资者
- 交易所级且对所有投资者

9.1.5 风险警示板买入量风控

风险警示板买入量风控可对投资者的风险警示板日买入量进行控制，控制对象为单个风险警示板证券的日买入量，一旦触发了风控阈值，该风险警示板证券无法继续发送买入指令。

本风控并非在买入量到达阈值以后才开始风控，否则新的买入报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能买入量，当买入报单报出去以后易达就会增加可能买入量，处于挂单状态时不改变可能买入量，报单状态达到终态后从可能买入量中扣减未成交买入量。因此，如果当前有大量买入挂单也会导致后续买入报单因为本风控而被拒绝。

风险警示板买入量风控参数如下表所示：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_ExchangeSTBuyVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDExchange.ExchangeRef，唯一确定交易所
	IntValue1	最大日买入量
产品级	GeneralRiskParamType	固定为YD_GRPT_ProductSTBuyVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDProduct.ProductRef，唯一确定产品
	IntValue1	最大日买入量

层级	字段	说明
合约级	GeneralRiskParamType	固定为YD_GRPT_ExchangeBuyVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDInstrument.InstrumentRef，唯一确定合约
	IntValue1	最大日买入量

由于各个层级中AccountRef可以为全体账户或者指定账户两种，因此两两组合一共六个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 合约级且指定账户
- 产品级且指定账户
- 交易所级且指定账户
- 合约所级且对所有投资者
- 产品所级且对所有投资者
- 交易所级且对所有投资者

9.1.6 现货持仓量风控

现货持仓量风控可对投资者的现货持仓量进行控制，控制对象为单个现货证券的持仓量，一旦触发了风控阈值，该现货证券无法继续发送买入指令。

本风控并非在持仓量到达阈值以后才开始风控，否则新的买入报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能持仓量，当买入报单报出去以后易达就会增加可能持仓量，处于挂单状态时不改变可能持仓量，报单状态达到终态后从可能持仓量中扣减未成交持仓量。因此，如果当前有大量买入挂单也会导致后续买入报单因为本风控而被拒绝。

现货持仓量风控参数如下表所示：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_ExchangeHoldingLimit
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDExchange.ExchangeRef，唯一确定交易所
	IntValue1	最大持仓量
产品级	GeneralRiskParamType	固定为YD_GRPT_ProductHoldingLimit
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDProduct.ProductRef，唯一确定产品
	IntValue1	最大持仓量
合约级	GeneralRiskParamType	固定为YD_GRPT_InstrumentHoldingLimit
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDInstrument.InstrumentRef，唯一确定合约

层级	字段	说明
	IntValue1	最大持仓量

由于各个层级中AccountRef可以为全体账户或者指定账户两种，因此两两组合一共六个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 合约级且指定账户
- 产品级且指定账户
- 交易所级且指定账户
- 合约所级且对所有投资者
- 产品所级且对所有投资者
- 交易所级且对所有投资者

9.1.7 交易所请求速度风控

限制投资者每秒向交易所发送的请求的总和，柜台采用滑动窗口统计交易所请求，不按照整秒统计。交易所请求包括所有通过柜台的钱仓、风控检查后，发给交易所的普通报单及其撤单请求，不包含其他报单（例如组合持仓、行权等）、报价和撤报价，也不包括被柜台拦截的请求。

交易所请求速度风控参数如下表所示：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_ExchangeRequestSpeed
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDExchange.ExchangeRef，唯一确定交易所
	IntValue1	每秒限速

9.1.8 交易所请求总数风控

限制投资者每日向交易所发送的请求的总和。交易所请求包括所有通过柜台的钱仓、风控检查后，发给交易所的普通报单及其撤单请求，不包含其他报单（例如组合持仓、行权等）、报价和撤报价，也不包括被柜台拦截的请求。

本风控的风控对象为整个账户向交易所发送的请求总和，可设置多个阈值，以控制不同指令类型的数量：

- 报单时的总数上限：当发送报单时，账户交易所请求总数超过本阈值，则拒绝该指令。
- 撤单时的总数上限：当发送撤单时，账户交易所请求总数超过本阈值，则拒绝该指令。

例如，设置报单时的总数上限为29000，撤单时的总数上限为30000，那么在交易所请求总数低于29000时，可以正常报撤单；当交易所请求总数高于29000但低于30000时，只能撤单不能报单；当交易所请求总数到达30000时，报撤单都被拒绝。

交易所请求总数风控参数如下表所示：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_MaxExchangeRequest
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身

层级	字段	说明
	ExtendedRef	对应于YDExchange.ExchangeRef，唯一确定交易所
	IntValue1	报单时的总数上限
	IntValue2	撤单时的总数上限

9.1.9 撤单笔数风控

撤单笔数风控分为产品层和合约层两个维度。

产品层的风控规则是对属于该产品的所有合约的特定报单的撤单笔数加总进行汇总风控，一旦触发了产品层风控阈值，该产品内所有合约都无法继续发送任何特定报单指令。

合约层的风控规则是针对单个合约的特定报单的撤单笔数加总后进行风控，一旦触发了合约层风控阈值，该合约无法继续发送任何特定报单指令。

上述特定报单的定义如下：

- 中金所国债期货的FAK和FOK报单
- 中金所除国债期货外的限价单
- 上期所、能源所、大商所、郑商所、广期所的限价单

本风控并非在特定报单的撤单笔数到达阈值以后才开始风控，否则新的报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能撤单笔数，当特定报单报出去以后易达就会增加可能撤单笔数，处于挂单状态时或者最终撤单则不改变可能撤单笔数，若全部成交则将撤单笔数扣减一笔。因此，如果当前有大量特定挂单也会导致后续特定报单因为本风控而被拒绝。

产品层的各个风控参数可以参考YDAccountProductInfo.TradingConstraints[HedgeFlag]的CancelLimit，合约层的各个风控参数可以参考YDAccountInstrumentInfo.TradingConstraints[HedgeFlag]的CancelLimit，具体字段含义请参考风控参数。

9.1.10 单笔报单量风控

虽然交易所控制了每个合约的最大报单量，但是对于某些投资者的某些策略还是偏大，需要由柜台协助控制，为此易达支持限制每笔报单的最大下单量。

报单量风控的参数在通用风控参数中，分别支持在交易所、产品以及合约三个维度上进行设置：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_ExchangeMaxOrderVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDExchange.ExchangeRef，唯一确定交易所
	IntValue1	报单量上限
产品级	GeneralRiskParamType	固定为YD_GRPT_ProductMaxOrderVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDProduct.ProductRef，唯一确定产品
	IntValue1	报单量上限

层级	字段	说明
合约级	GeneralRiskParamType	固定为YD_GRPT_InstrumentMaxOrderVolume
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	对应于YDInstrument.InstrumentRef，唯一确定合约
	IntValue1	报单量上限

由于各个层级中AccountRef可以为全体账户或者指定账户两种，因此两两组合一共六个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 合约级且指定账户
- 产品级且指定账户
- 交易所级且指定账户
- 合约所级且对所有投资者
- 产品所级且对所有投资者
- 交易所级且对所有投资者

9.1.11 价格偏离度风控

当报单价格与动态基准价之差超过阈值，或者当报单价格与最新价之差超过阈值，该报单会被柜台拒绝。其中，动态基准价在不同交易所的取值规则不同：

- 对于上交所和深交所，动态基准价是指合约在最近一次集合竞价阶段产生的成交价格，开盘集合竞价阶段未产生成交价格的，以前结算价格作为最近参考价格；盘中集合竞价阶段未产生成交价格的，以进入该集合竞价阶段前的最后一笔成交价格作为最近参考价格
- 对于期货交易所，若行情中的成交量不为0时，动态基准价取最新价，否则取昨结算价

易达在行情中直接提供了动态基准价YDMarketData.DynamicBasePrice。若柜台上没有配置价格偏离度风控，则动态基准价始终为昨结算价。

本风控规则中涉及到大量参数，满足以下任何一条均会触发风控规则：

- 报单价 $\geq$ 动态基准价 $\times$ $(1 +$ 动态基准价偏离比例上限$)$
- 报单价 $\leq$ 动态基准价 $\times$ $(1 -$ 动态基准价偏离比例下限$)$
- 报单价 $\geq$ 动态基准价 $\times$ $(1 +$ 动态基准价偏离$tick$数上限$)$
- 报单价 $\leq$ 动态基准价 $\times$ $(1 -$ 动态基准价偏离$tick$数下限$)$
- 报单价 $\geq$ 最新价 $\times$ $(1 +$ 最新价偏离比例上限$)$
- 报单价 $\leq$ 最新价 $\times$ $(1 -$ 最新价偏离比例下限$)$
- 报单价 $\geq$ 最新价 $\times$ $(1 +$ 最新价偏离$tick$数上限$)$
- 报单价 $\leq$ 最新价 $\times$ $(1 -$ 最新价偏离$tick$数下限$)$

由于字段限制，易达需要通过多条配置为相同产品的通用风控规则合成表示一条本风控规则。本风控规则只允许配置在全局或者产品层面，即ExtendedRef设置为空表示对所有产品生效，设置产品名表示仅对该产品生效，且不允许指定账户，因此下文中的AccountRef和ExtendedRef均被略去。请注意，有可能同时收到ExtendedRef为空和设置了个别产品的多组参数，请注意分辨。价格偏离度的其余参数如下表所示：

层级	字段	说明
动态基准价偏离比例上限	GeneralRiskParamType	固定为YD_GRPT_DynamicPriceLimitUpperRatio

层级	字段	说明
	FloatValue	比例阈值
动态基准价偏离比例下限	GeneralRiskParamType	固定为YD_GRPT_DynamicPriceLimitLowerRatio
	FloatValue	比例阈值
动态基准价偏离tick数上限	GeneralRiskParamType	固定为YD_GRPT_DynamicPriceLimitUpperTickCount
	IntValue1	tick数
动态基准价偏离tick数下限	GeneralRiskParamType	固定为YD_GRPT_DynamicPriceLimitLowerTickCount
	IntValue1	tick数
最新价偏离比例上限	GeneralRiskParamType	固定为YD_GRPT_DynamicLastPriceLimitUpperRatio
	FloatValue	比例阈值
最新价偏离比例下限	GeneralRiskParamType	固定为YD_GRPT_DynamicLastPriceLimitLowerRatio
	FloatValue	比例阈值
最新价偏离tick数上限	GeneralRiskParamType	固定为YD_GRPT_DynamicLastPriceLimitUpperTickCount
	IntValue1	tick数
最新价偏离tick数下限	GeneralRiskParamType	固定为YD_GRPT_DynamicLastPriceLimitLowerTickCount
	IntValue1	tick数

由于ExtendedRef可以为全部产品和指定产品两种，因此一共两个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 指定产品
- 全部产品

9.1.12 期权买入额度

本风控规则可以控制期权买入额度（期权多头持仓总成本）的规模，期权多头的持仓成本是把相应维度上（某账户全部或者某账户在某交易所上）期权多头持仓的持仓成本相加得到的，单条期权多头的持仓成本为其成交明细成本的总和，
$$\text{成交明细成本} = \text{成交价格} \times \text{交易所保证金率} \times \text{期权合约乘数} \times \text{成交量}$$。

本风控规则支持以下子规则：

- 汇总买入额度风控，即将所有交易所买入额度加总后予以控制，只要该汇总买入额度超限，所有交易所的报单都会受到限制
- 单交易所买入额度风控，即只汇总单个交易所的买入额度，只有当某交易所的买入额度超限后才会对该交易所的报单会受到限制

汇总买入额度风控参数如下表所示：

层级	字段	说明
全局级	GeneralRiskParamType	固定为YD_GRPT_OptionLongPositionCost
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身

层级	字段	说明
	FloatValue	持仓成本限额阈值

如果ydExtendedApi，则可以通过YDExtendedAccount.OptionLongPositionCostLimit查到该投资者全局层面的持仓成本阈值，也可以通过YDExtendedAccount.OptionLongPositionCost查到该投资者全局层面的当前持仓成本汇总值。还可以通过YDExtendedAccountExchangeInfo.OptionLongPositionCostLimit查到该投资者在交易所层面的持仓成本阈值，也可以通过YDExtendedAccountExchangeInfo.OptionLongPositionCost查到该投资者在交易所层面的当前持仓成本汇总值。

由于AccountRef可以为全体账户或者指定账户两种，因此一共两个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 指定账户
- 对所有投资者

单交易所买入额度风控参数如下表所示：

层级	字段	说明
交易所级	GeneralRiskParamType	固定为YD_GRPT_ExchangeOptionLongPositionCost
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	可以设置为空表示对所有交易所生效，也可以设置某交易所的ExchangeRef表示仅对设置交易所生效，ExchangeRef可以在YDExchange中获取
	FloatValue	持仓成本限额阈值

由于AccountRef可以为全体账户或者指定账户两种，ExtendedRef可以为全部交易所或者指定交易所两种，因此一共四个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 指定账户，指定交易所
- 指定账户，全部交易所
- 全部账户，指定交易所
- 全部账户，全部交易所

9.1.13 事前信息量风控

为帮助投资者事前控制信息量上限，完善事后信息量风控在控制力度上的不足，易达将根据最大信息量进行事前风控，若报单将超过最大信息量上限，该报单将被柜台拒绝。信息量的计算方法参考期货信息量申报费。

> 无法提供基于OTR的信息量上限事前风控的原因是，在达到更高OTR等级后，投资者可以下更多有成交的报单减小OTR。如果到达OTR后禁止报单，那么将断绝投资者自行调整的可能性。而信息量是单调递增的数字，因此可以限制其最大值。

只有当投资者在该合约上收取信息量申报费，且YDAccountInstrumentInfo.MaxMessage大于0时，才会控制该投资者在该合约上的最大信息量。与其他事前风控类似，本风控并非在报单的信息量到达阈值以后才开始风控，否则新的报单一旦报到交易所就会超过风控阈值。因此，易达实际上控制的是可能信息量，当报单报出以后易达就会增加可能信息量，若撤单则不改变可能信息量，若全部成交则减少可能信息量。因此，如果当前有大量挂单也会导致后续报单因为本风控而被拒绝。

与其他事前风控不同，该风控设置直接从CTP日初数据中转换而来，只要在CTP中设置投资者最大信息量风控，那么易达中将自动设置该风控规则。与CTP类似，易达也支持盘中修改最大信息量，如需盘中调整请联系管理员设置。

若盘中经纪商调整了最大信息量风控参数，API会通过下述回调函数通知投资者：

```
virtual void notifyUpdateMessageCommissionConfig(const YDUpdateMessageCommissionConfig *pUpdateMessageCommissionConfig)
```

其中，YDUpdateMessageCommissionConfig的字段信息如下：

字段	说明
AccountRef	账户编号。可从YDAccount获取。
ExchangeRef	交易所编号。可从YDExchange获取。
ProductRef	产品编号。可从YDProduct获取。
InstrumentRef	合约编号。可从YDInstrumentRef获取。
MaxMessage	更新后的最大信息量阈值

9.1.14 信息量手续费风控

为帮助投资者控制其信息量手续费上限，易达提供了本事前风控。本风控只适用于普通报单，不适用于报价、其他类型的报单（例如行权、组合持仓等）和管理员发送的指令。

本风控的风控对象为整个账户的信息量手续费总额，可设置多个阈值，以控制不同指令类型的数量：

- 普通开仓报单阈值：当收到普通开仓报单时刻的账户信息量手续费超过本阈值，则拒绝该指令。组合合约报单时只要有一腿是开仓就使用开仓报单阈值。
- 普通平仓报单阈值：当收到普通平仓报单时刻的账户信息量手续费超过本阈值，则拒绝该指令。
- 撤普通报单阈值：当收到撤普通报单时刻的账户信息量手续费超过本阈值，则拒绝该指令。

当指令因为本风控而被拒绝时，返回的ErrorNo为YD_ERROR_MessageCommissionExceed=115。

由于本风控只在收到指令时检查当前的信息量手续费，并没有预冻结信息量手续费，因此实际值可能略微超过设置上限。

信息量手续费风控参数如下表所示：

层级	字段	说明
全局级	GeneralRiskParamType	固定为YD_GRPT_MaxMessageCommission
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	FloatValue	普通开仓报单阈值，设置值为0表示无限制
	IntValue1	普通平仓报单阈值，设置值为0表示无限制
	IntValue2	撤普通报单阈值，设置值为0表示无限制

9.2 事后风控

9.2.1 事后开仓量风控

当该期权系列的总买开仓量大于等于最大买开仓量，或者卖开仓量大于等于最大卖开仓量时，对该账户该期权系列的所有合约设置只平权限。

易达事后风控系统支持以下六类风控：

- 单合约开仓量限制：限制某合约开仓量

- 产品下单合约开仓量限制：对于指定产品下的每个合约，限制单个合约开仓量
- 交易所下单合约开仓量限制：对于指定交易所下的每个合约，限制单个合约开仓量
- 期权系列汇总开仓量限制：限制指定期权系列下所有合约的加总开仓量
- 产品汇总开仓量限制：限制指定产品下所有合约的加总开仓量
- 交易所下产品汇总开仓量限制：对于指定交易所下的每个产品， 限制单个产品下所有合约的加总开仓量

本风控规则的参数如下：

层级	字段	说明
单合约开仓量限制	GeneralRiskParamType	固定为1019
	AccountID	资金账号
	ExtendedID	合约
	IntValue1	最大买开仓量
	IntValue2	最大卖开仓量
产品下单合约开仓量限制	GeneralRiskParamType	固定为1020
	AccountID	资金账号
	ExtendedID	产品
	IntValue1	最大买开仓量
	IntValue2	最大卖开仓量
交易所下单合约开仓量限制	GeneralRiskParamType	固定为1021
	AccountID	资金账号
	ExtendedID	交易所
	IntValue1	最大买开仓量
	IntValue2	最大卖开仓量
期权系列汇总开仓量限制	GeneralRiskParamType	固定为1007
	AccountID	资金账号
	ExtendedID	可以是以下两种格式： 期权产品代码:期权到期年份:期权到期月份 期权产品代码:基础合约代码
	IntValue1	最大买开仓量
	IntValue2	最大卖开仓量
产品汇总开仓量限制	GeneralRiskParamType	固定为1022
	AccountID	资金账号
	ExtendedID	产品
	IntValue1	最大买开仓量
	IntValue2	最大卖开仓量

层级	字段	说明
交易所下产品汇总开仓量限制	GeneralRiskParamType	固定为1023
	AccountID	资金账号
	ExtendedID	交易所
	IntValue1	最大买开仓量
	IntValue2	最大卖开仓量

本风控规则的不同类型之间没有覆盖关系，若同时在多个层级设置风控规则，则这些风控规则同时生效。

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

9.2.2 事后成交量风控

当该期权系列的总成交量大于等于最大成交量时，对该账户该期权系列的所有合约设置只平权限。

易达事后风控系统支持以下六类风控：

- 单合约成交量限制：限制某合约成交量
- 产品下单合约成交量限制：对于指定产品下的每个合约，限制单个合约成交量
- 交易所下单合约成交量限制：对于指定交易所下的每个合约，限制单个合约成交量
- 期权系列汇总成交量限制：限制指定期权系列下所有合约的加总成交量
- 产品汇总成交量限制：限制指定产品下所有合约的加总成交量
- 交易所下产品汇总成交量限制：对于指定交易所下的每个产品， 限制单个产品下所有合约的加总成交量

本风控规则的参数如下：

层级	字段	说明
单合约成交量限制	GeneralRiskParamType	固定为1014
	AccountID	资金账号
	ExtendedID	合约
	IntValue1	最大成交量
产品下单合约成交量限制	GeneralRiskParamType	固定为1015
	AccountID	资金账号
	ExtendedID	产品
	IntValue1	最大成交量
交易所下单合约成交量限制	GeneralRiskParamType	固定为1016
	AccountID	资金账号
	ExtendedID	交易所
	IntValue1	最大成交量
期权系列汇总成交量限制	GeneralRiskParamType	固定为1006
	AccountID	资金账号

层级	字段	说明
	ExtendedID	可以是以下两种格式： 期权产品代码:期权到期年份:期权到期月份 期权产品代码:基础合约代码
	IntValue1	最大成交量
产品汇总成交量限制	GeneralRiskParamType	固定为1017
	AccountID	资金账号
	ExtendedID	产品
	IntValue1	最大成交量
交易所下产品汇总成交量限制	GeneralRiskParamType	固定为1018
	AccountID	资金账号
	ExtendedID	交易所
	IntValue1	最大成交量

本风控规则的不同类型之间没有覆盖关系，若同时在多个层级设置风控规则，则这些风控规则同时生效。

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

9.2.3 事后持仓量风控

当该期权系列的多头总持仓大于等于多头限仓，或者空头总持仓大于等于空头限仓时，对该账户该期权系列的所有合约设置只平权限。如果之后通过平仓，使总持仓小于限仓了，本规则不会自动设置允许交易权限，但是如果管理员手工将其设置为允许交易权限，本系统也不会再次将其改为只平权限。

易达事后风控系统支持以下六类风控：

- 单合约持仓量限制：限制某合约持仓量
- 产品下单合约持仓量限制：对于指定产品下的每个合约，限制单个合约持仓量
- 交易所下单合约持仓量限制：对于指定交易所下的每个合约，限制单个合约持仓量
- 期权系列汇总持仓量限制：指定期权系列的合约中，所有看涨期权的买持仓量和看跌期权的卖持仓量之和不超过多头限仓，看跌期权的买持仓量和看涨期权的卖持仓量之和不超过空头限仓
- 产品汇总持仓量限制：指定期货产品的合约中，期货多头持仓量之和不超过多头限仓，期货空头持仓量之和不超过空头限仓；指定期权产品的合约中，所有看涨期权的买持仓量和看跌期权的卖持仓量之和不超过多头限仓，看跌期权的买持仓量和看涨期权的卖持仓量之和不超过空头限仓
- 交易所下产品汇总持仓量限制：对于指定交易所下的每个产品，限制单个产品的多空限仓，计算规则同上述“产品汇总持仓量限制”

> **Note**
>
> 从1.486.96.57开始，易达根据交易所规则修订了期权多空持仓的汇总规则，若需使用期权的事后持仓量风控，则需确保柜台版本升级至上述版本或以上版本。

本风控规则的参数如下：

层级	字段	说明
单合约持仓量限制	GeneralRiskParamType	固定为1009
	AccountID	资金账号

层级	字段	说明
	ExtendedID	合约
	IntValue1	多头限仓
	IntValue2	空头限仓
产品下单合约持仓量限制	GeneralRiskParamType	固定为1010
	AccountID	资金账号
	ExtendedID	产品
	IntValue1	多头限仓
	IntValue2	空头限仓
交易所下单合约持仓量限制	GeneralRiskParamType	固定为1011
	AccountID	资金账号
	ExtendedID	交易所
	IntValue1	多头限仓
	IntValue2	空头限仓
期权系列汇总持仓量限制	GeneralRiskParamType	固定为1005
	AccountID	资金账号
	ExtendedID	可以是以下两种格式： 期权产品代码:期权到期年份:期权到期月份 期权产品代码:基础合约代码
	IntValue1	多头限仓
	IntValue2	空头限仓
产品汇总持仓量限制	GeneralRiskParamType	固定为1012
	AccountID	资金账号
	ExtendedID	产品
	IntValue1	多头限仓
	IntValue2	空头限仓
交易所下产品汇总持仓量限制	GeneralRiskParamType	固定为1013
	AccountID	资金账号
	ExtendedID	交易所
	IntValue1	多头限仓
	IntValue2	空头限仓

本风控规则的不同类型之间没有覆盖关系，若同时在多个层级设置风控规则，则这些风控规则同时生效。

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

9.2.4 交易所错单笔数风控

交易所对错单笔数是有监管的，但是监管措施较为灵活，为了避免这种情况的发生，经纪商通常希望能控制投资者的交易所错单笔数，而且即便稍微超过一点也问题不大，因此，易达使用事后风控来控制交易所错单笔数，在达到错单笔数阈值后，易达系统会将该客户的交易权限设置为禁止交易。请注意只有从交易所返回的错单才增加错单笔数总数，被柜台拦截的错单不会增加错单笔数。

本风控规则在计算交易所错单时是不精确的，如果发生了ydServer的重启，而客户端没有使用高可用模式恢复，那么前一次ydServer运行时发生的错单将不被计数。

全局级的风控参数如下表所示：

层级	字段	说明
全局级	GeneralRiskParamType	固定为1004
	AccountID	资金账户
	FloatValue	错单笔数阈值

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

9.2.5 成交持仓比风控

当某产品内所有合约的成交量达到一定规模时，则该产品的成交量/持仓量的比值不能超过一定比值，否则会被视为违规，当触发风控规则后，易达会向经纪商的业务人员发出警告以引起其关注，并不会有刚性的风控措施。本风控规则只对上交所和深交所的期权有效。本风控会设置到柜台上，风控参数也会下发到API。

成交量为某产品的所有合约上买卖开平的成交的成交量的总数。

持仓量为某产品的所有合约的上一交易日结算后持仓量总和与当前交易日结算后持仓量总和的大者，即$max(某产品的所有合约上一交易日结算后持仓量总和，某产品的所有合约当前交易日结算后持仓量总和)$。

上一交易日结算后持仓量总和即为当日的日初持仓，而当前交易日结算后持仓量总和则需要在盘中根据结算规则推算，具体规则为：

1. 同一合约的除组合仓之外的多空持仓在结算时会对冲，因此单个合约结算后持仓量的计算公式为：$|多头可用持仓量 - 空头可用持仓量| + 多头冻结量 + 空头冻结量$。其中多头和空头可用持仓量为当前持仓量扣除平仓挂单冻结量和组合持仓量之后的值，冻结量为平仓挂单冻结和组合持仓量之和
2. 对于当天到期的合约，将多空持仓全部解组合后，按照第1条中的方法计算结算后持仓量
3. 对于距离到期日剩余2天的合约，将认购牛市价差、认购熊市价差、认沽牛市价差、认沽熊市价差类型的组合全部解组后，按照第1条中的方法计算结算后持仓量

> 假设某投资者的持仓和当日交易情况如下：
>
> - 当天没有合约到期或者距离到期日2天
> - A合约的日初权利仓4手，当日开权利仓11手，开义务仓5手
> - B合约有日初义务仓2手，当日开权利仓3手，开义务仓7手
> - 建立1手认购牛市价差组合，第一腿为A的权利仓，第二腿为B的义务仓
>
> 则计算持仓量的过程如下：
>
> 上一交易日结算后持仓=4+2=6
>
> 当前交易日结算后持仓A=|(4+11-1)-5|+1=10，其中括号内的减一是组合仓位
>
> 当前交易日结算后持仓B=|3-(2+7-1)|+1=6，其中括号内的减一是组合仓位
>
> 当前交易日结算后持仓总和=10+6=16

> 持仓量=max(6, 16)=16
>
> 成交量=11+5+3+7=26
>
> 成交持仓比=26/16=1.625

风控参数如下表所示：

层级	字段	说明
产品级	GeneralRiskParamType	固定为YD_GRPT_TradePositionRatio
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	可以设置为空表示对所有产品生效，也可以设置单一产品的ProductRef表示仅对设置产品生效，ProductRef可以在YDProduct中获取
	IntValue1	成交量规模阈值，当总成交量达到阈值时才进行成交持仓比的计算
	FloatValue	成交持仓比阈值

由于AccountRef可以为全体账户或者指定账户两种，ExtendedRef可以为全部交易所或者指定交易所两种，因此一共四个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 指定账户，指定交易所
- 指定账户，全部交易所
- 全部账户，指定交易所
- 全部账户，全部交易所

目前本风控规则由ydClient负责执行，投资者可以通过ydClient查看风控参数与风险规则状态，也可以通过API获取风控参数。

9.2.6 撤单报单比风控

当某产品内所有合约的报单笔数之和达到一定规模时（假设为X），则该产品的(撤单笔数-X)/(报单笔数-X)的比值不能超过一定比值，否则会被视为违规，当触发风控规则后，易达会向经纪商的业务人员发出警告以引起其关注，并不会有刚性的风控措施。其中，撤单笔数为属于该产品的所有合约上的撤单笔数之和，报单笔数为属于该产品的所有合约上的报单笔数之和。本风控会设置到柜台上，风控参数也会下发到API。

风控参数如下表所示：

层级	字段	说明
产品级	GeneralRiskParamType	固定为YD_GRPT_OrderCancelRatio
	AccountRef	-1表示对所有投资者生效，否则表示用户的AccountRef值，对于投资者账户登录来说就是该账户本身
	ExtendedRef	可以设置为空表示对所有产品生效，也可以设置单一产品的ProductRef表示仅对设置产品生效，ProductRef可以在YDProduct中获取
	IntValue1	报单笔数阈值，当总报单笔数达到阈值时才进行撤单报单比的计算
	FloatValue	撤单报单比阈值

由于AccountRef可以为全体账户或者指定账户两种，ExtendedRef可以为全部交易所或者指定交易所两种，因此一共四个层级，他们优先级从高到低（高优先级覆盖低优先级的参数配置）分别是：

- 指定账户，指定交易所
- 指定账户，全部交易所
- 全部账户，指定交易所
- 全部账户，全部交易所

目前本风控规则由ydClient负责执行，投资者可以通过ydClient查看风控参数与风险规则状态，也可以通过API获取风控参数。

9.2.7 事后信息量风控

目前上期所、大商所、郑商所都放弃了硬性的撤单数控制转而使用信息量控制投资者的报撤单数量，以减少无效报单对交易所的压力。不少投资者对此转变并不适应，甚至投资者可能因为没有控制好报单量导致支付了高额的信息量手续费。虽然主流主席柜台和易达柜台均已支持信息量申报费扣减和信息量事前风控功能，但是由于事后风控的灵活性，部分投资者和经纪商仍然偏好使用事后信息量风控，因此该风控功能予以保留。信息量和OTR（报单成交比）的计算方式请参考衍生品信息量申报费。

易达提供了两种不同的信息量控制方式，分别是单信息量控制和信息量与OTR共同控制。

单信息量控制时，只要信息量超过设定的阈值就会按照不同的阈值设置投资者在该合约上面的交易权限为只能平仓或者禁止交易。易达提供了与单信息量事后风控对应的事前风控事前信息量风控，在使用单信息量风控时可以优先考虑使用事前信息量风控。单信息量控制的风控参数如下，这些参数在外部风控程序中设置，目前不会下发到API：

层级	字段	说明
产品级	GeneralRiskParamType	固定为1001
	AccountID	资金账号
	ExtendedID	对应于YDProduct.ProductID，唯一确定产品
	IntValue1	信息量阈值，触发后设置该合约只能平仓权限
	IntValue2	信息量阈值，触发后设置该合约禁止交易权限

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

信息量与OTR共同控制时，只有当OTR和信息量都超过设定的阈值时，才会按照不同的阈值设置投资者在该合约上面的交易权限为只能平仓或者禁止交易。其风控参数如下：

层级	字段	说明
产品级	GeneralRiskParamType	固定为1002
	AccountID	资金账号
	ExtendedID	对应于YDProduct.ProductID，唯一确定产品
	IntValue1	信息量阈值，触发后设置该合约只能平仓权限
	IntValue2	信息量阈值，触发后设置该合约禁止交易权限
	FloatValue	OTR阈

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

9.2.8 临近到期日禁止开仓

为方便经纪商为临近到期日的合约设置禁止开仓的交易权限，易达借助事后风控规则实现了便捷的设置方法。

本风控规则会根据YDInstrument.ExpireTradingDayCount判断是否到达了设定的阈值，若到达则会相应设置各个投资者上对应合约的只能平仓交易权限。

本风控规则的参数如下：

层级	字段	说明
产品级	GeneralRiskParamType	固定为1003
	AccountID	资金账号
	ExtendedID	对应于YDProduct.ProductID，唯一确定产品
	IntValue1	距离到期日的天数阈值

目前本风控规则由外部风控系统负责执行，因此投资者无法通过API获取具体风控参数值，也无法获知风控规则是否触发。

10 中继

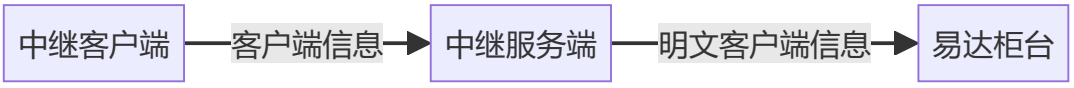
易达支持管理员用户登录的中继模式，中继模式下易达API的使用方式与普通投资者直接接入基本一致，以下是中继系统接入易达柜台的额外工作：

- 创建特殊的管理员账户：用于中继系统登录的管理员账户必须拥有`中继Relay`权限，一旦管理员用户被设置了`中继Relay`权限，那么其他管理员权限均被禁止
- 上报客户端信息：由于证券公司和期货公司分属投保中心和保证金监控中心监管，因此当中继系统需要根据部署经纪商类型使用相应的客户端信息上报方式
- 指定账户：易达API的接口上均有pAccount参数用于指定投资者，普通投资者交易时可以设置为NULL表示使用当前投资者，但中继系统必须在API指令中填写pAccount参数以标识该指令的投资者。

> 中继系统服务端连接柜台总体分为三种模式，分别是：
>
> - 使用管理员用户登录柜台，中继服务端只建立唯一一个到柜台的连接，所有投资者均使用该连接报入柜台
> - 使用投资者账号登录柜台，中继服务端为每个中继客户端建立一个到柜台的客户端专属连接，中继客户端的报单通过客户端专属连接报入柜台
> - 使用投资者账号登录柜台，中继服务器为每个投资者最多建立一个到柜台的投资者专属连接，投资者的报单均通过投资者专属连接报入柜台
>
> 目前易达只支持管理员用户登录模式，由于管理员登录只能使用TCP报单，因此相比UDP或者XTCP报单穿透性能有所下降（5-6微秒）。考虑到中继交易模式对性能并没有极致的要求，因此管理员模式应该可以满足大部分中继系统的要求，如算法平台等。若对性能有较高要求而必须使用UDP或者XTCP报单的，请与我们联系。

10.1 上报证券客户端信息

根据《证券公司客户交易终端信息管理技术规范》要求，终端信息字符串应以满足向监管机构报送要求为前提，存储于集中交易系统、统一认证系统和日志系统等系统。中继服务器应将投资者的信息已明文的方式上报易达柜台，易达柜台将客户端信息写入盘后下场数据，供集中交易系统盘后加载保存。



为满足各证券公司的差异化需求，易达将为每种需求提供专属的终端信息上报方法。目前只提供了以下一种上报方法。

由于终端信息需和报单一一对应，因此中继服务器在每笔报撤单前都应上报对应的终端信息，如果连续多笔报撤单来自于同一个终端的同一投资者，那么除了第一笔报撤单前需要上报外，剩余报撤单向柜台发送前无需重复上报终端信息，易达柜台会使用最近一次终端报送结果作为后续报单的终端信息。

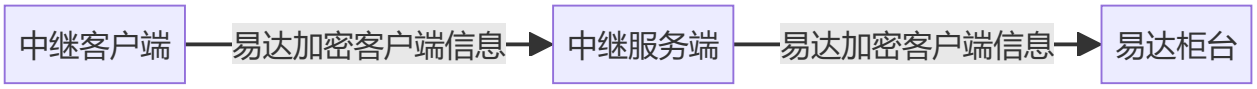
```
1 virtual bool setRelayContext1(const char *clientInfo,int loginOption)
```

clientInfo为明文终端信息，最长支持256字节。易达柜台不会对传入的明文终端信息做任何处理，因此，中继服务器必须保证传入的终端信息符合监管格式要求，包括改写转义字符。

loginOption为投资者登录的操作方式（也被称为operway），最长支持4个字节的操作方式。若该操作方式不在投资者被允许的操作方式范围内，后续该投资者上报的报单会被柜台拒绝，并通过notifyOrder返回ErrorNo为YD_ERROR_InvalidLoginInfo的错单。

10.2 上报期货客户端信息

根据保证金监控中心监管规范，部署在期货公司的中继客户端必须使用柜台开发商提供的客户端采集工具生成加密的客户端信息，此加密客户端信息将经由中继服务端上报给柜台。其过程如下图所示。



10.2.1 中继客户端信息采集

易达在ydClient包中提供了win64和linux64两个平台的客户端采集工具ydSysInfo，请根据客户端所在平台选择使用。

```
/// If encrypedSysInfo is not NULL, it should be a buffer at least 1024 bytes. It can be used for relay mode application
/// Return value indicates problem in sysInfo. NULL means no problem. If there are multiple problems, only one of them will be returned.
YD_SYS_INFO_API_EXPORT const char *genClientSysInfo(char *encryptedSysInfo)
```

encryptedSysInfo为投资者创建的字符数组，至少应为1024字节，用于存储从中继客户端获取的易达加密客户端信息。若该方法返回NULL，则表示信息才正常，若该方法返回值不为NULL，则表示部分信息采集失败，投资者可查看字符串返回值以获得采集失败提示信息，如需获取完整的明文信息，请联系经纪商在柜台查看。采集完成后，中继客户端应将此易达加密客户端信息发送到中继服务器。

10.2.2 中继服务端信息上报

由于中继客户端可能将采集不完整的信息报送到中继服务器，中继服务器收到易达加密客户端信息后，可以使用以下方法校验加密信息是否完整。encryptedClientReport为中继客户端传入的密文信息，返回值规则与genClientSysInfo一致。

```
/// return error message, NULL if error
virtual const char *verifyClientInfo(const char *encryptedClientReport)
```

中继服务端可使用以下方法将中继客户端的加密信息发送到易达柜台。该方法的返回值表示密文信息是否发送成功，并不确保发送信息的完整性和正确性。

```
virtual bool reportRelayClientInfo(const YDRelayClientInfo *pRelayClientInfo)
```

YDRelayClientInfo的字段说明如下所示：

参数	字段	说明
YDRelayClientInfo	AccountRef	投资者资金账号序号，请参考获取投资者账户信息获取账户，并从YDAccount中获取AccountRef
	RequestID	请求编号，可以与notifyResponse中的RequestID对应
	EncryptedClientReport	中继客户端传入的密文信息
	ClientIP	中继客户端登录中继服务器的IP地址
	ClientPort	中继客户端登录中继服务器的端口
	ClientLoginTime	中继客户端登录中继服务器的登录时间字符串，例如10:20:00
	ClientAppID	中继客户端登录中继服务器的AppID

柜台会检查中继服务器传入的中继客户端密文信息是否存在错误或者缺失，并通过notifyResponse返回检查结果：

参数	字段	说明
YDRelayClientInfo	errorNo	YD_ERROR_NotRelaySession：非中继连接，不可以报送 YD_ERROR_InvalidRelayClientInfo：中继客户端加密信息存在错漏
	requestType	固定为YD_RT_RelayClientInfo
	requestID	reportRelayClientInfo中传入的RequestID

10.3 获取投资者账户信息

管理员账户可通过以下方法获取所有投资者的账户：

```
virtual int getAccountCount(void)
virtual const YDAccount *getAccount(int pos)
virtual const YDAccount *getAccountByID(const char *accountID)
```

前两个方法可以遍历所有账户，具体方法如下面代码所示：

```
for(int i = 0; i < pApi->getAccountCount(); i++) {
  YDAccount *account = pApi->getAccount(i);
}
```

最后一个方法可以查找指定账户，调用示例如下所示：

```
pApi->getAccountByID("001")
```

11 管理

11.1 导出投资者数据

投资者和管理员可以使用以下方法导出任意时刻的投资者数据。dir为导出数据的目录，accountIDs是导出资金账户的列表，用空格分隔。投资者调用请保持accountIDs为空字符串表示导出本人数据，管理员调用时可以填入要导出的投资者列表，也可以填入空字符串表示导出所有投资者数据。

```
virtual bool exportData(const char *dir,const char *accountIDs="")
```

导出目录下会产生以下文件：

文件名	说明
instrument.csv	合约信息
rate.csv	衍生品保证金率和手续费率
cashCommissionRate.csv	现货交易规费率
tradingRight.csv	交易权限
account.csv	账户信息，包括各种资金信息
position.csv	衍生品持仓
holding.csv	现货持仓
combPosition.csv	组合持仓
marginSideInfo.csv	大边保证金信息
order.csv	报单
trade.csv	成交
quote.csv	报价

11.2 修改口令

盘中投资者可以通过API修改交易口令，修改后口令永久有效。调用方法如下：

```
virtual bool changePassword(const char *username,const char *oldPassword,const char *newPassword)
```

口令的修改结果会通过以下回调返回，errorNo为0表示修改成功，其他值表示修改错误，通常是因为旧口令错误YD_ERROR_OldPasswordMismatch或者不符合口令强度YD_ERROR_WeakPassword导致的。

```
virtual void notifyChangePassword(int errorNo)
```

11.3 写日志

易达API会默认往可执行程序路径下的log目录写入日志，投资者可以借用该机制写入自己的日志信息，若找不到log目录则不会产生日志文件。

```
virtual void writeLog(const char *format,...)
```

该日志是按天拆分的，通常API在启动的时候会产生以下日志样例。

```
14:59:54 YD API start
14:59:54 version 1.108.36.33
14:59:54 build time Mar  3 2022 17:37:32
14:59:54 build version GCC 10.2.0
14:59:54 TCP trading server 0 connected
```

11.4 出入金

易达提供了场上出入金功能，管理员用户可在盘中为投资者出入金和调整投资者的资金使用限度，出入金业务请参考出入金净额。本功能只能由拥有出入金权限的管理员用户调用。

```
virtual bool alterMoney(const YDAccount *pAccount,int alterMoneyType,double alterValue)
```

上述方法的参数说明如下：

参数	说明
YDAccount	待出入金的资金账户指针
alterMoneyType	出入金类型
alterValue	对应于出入金类型的数量

出入金类型及其含义如下表所示：

出入金类型	说明
YD_AM_ModifyUsage	调整投资者的资金使用限度。若调整值小于原有值，则会检查调整后是否会导致Available为负，若为负则不允许调整。Available的计算方法请参考可用资金，请注意此处的检查条件是Available，而不是可用资金useable()
YD_AM_Deposit	入金。在现有累计入金金额的基础上增加alterValue指定的金额。
YD_AM_FrozenWithdraw	冻结出金。在现有冻结出金金额的基础上增加alterValue指定的金额。
YD_AM_CancelFrozenWithdraw	撤销冻结出金。在现有冻结出金金额的基础上减少alterValue指定的金额。如果alterValue为0，表示撤销所有冻结出金。
YD_AM_TryWithdraw	先检查可用资金是否足够出金，若足够则出金，若不足则报错。
YD_AM_Withdraw	出金。在现有累计出金金额的基础上增加alterValue指定的金额。
YD_AM_DepositTo	入金到指定金额。直接设置现有累计入金金额，若调整值小于现有值，则不允许调整。
YD_AM_WithdrawTo	出金到指定金额。直接设置现有累计出金金额，若调整值小于现有值，则不允许调整。
YD_AM_ForceModifyUsage	强制调整投资者的资金使用限度，不检查调整后Available是否会变为负数。

> 易达有众多出入金相关工具，都将影响柜台最终的净出入金结果。我们建议经纪商仅选择使用其中的一种而不要混合使用，否则对这些工具同时作用于出入金时将难以分析净出入金结果。为了帮助投资者和经纪商分析和理解在某些特殊情况下使用了多种工具以后的影响，以下简要介绍各个工具对柜台出入金的影响方式：
>
> - ydClient和管理端的手工出入金：使用YD_AM_Deposit和YD_AM_Withdraw出入金

> - ydSync：从出入金源（CTPRisk、文件等）读取出入金流水，在ydSync中汇总后通过YD_AM_DepositTo和YD_AM_WithdrawTo出入金
> - ydFundManager和管理端的资金同步功能：从出入金源（CTPRisk、文件、手工操作等）读取出入金流水，在ydSync中汇总后通过YD_AM_DepositTo和YD_AM_WithdrawTo出入金

12 工具

12.1 ydcmd

ydcmd是供投资者使用的工具，主要解决在经纪商限制投资者使用YDClient的情况下的基本查询和应急处理的需求。目前支持批量撤单、批量撤报价、修改口令、导出数据。该命令的帮助信息如下，可以在命令行输入ydcmd获得。

```
ydcmd ydExport <config file> [<dir> [<username> [<password>]]]
ydcmd ydChangePassword <config file> [<username> [<password> [<new password>]]]
ydcmd ydCancelOrders <config file> [<username> [<password> [once]]]
ydcmd ydCancelQuotes <config file> [<username> [<password> [once]]]
```

ydcmd基于ydApi开发，其运行原理和特性与普通的API客户端一致，该工具的通用约定如下：

- 第一个参数始终是功能名。可将ydcmd文件名改为功能名后，直接通过功能名使用对应功能，后面的参数不变。例如将ydcmd改名为ydExport后，可以直接在命令行调用ydExport config.txt <dir> <username> <password>。
- 第二个参数始终是配置文件路径，配置方法与ydApi的配置文件要求完全一致，通常可以直接复用生产中的配置文件
- 用户名和口令有多种输入方式，若用户名和口令不在命令行指定，那么会在程序运行后提示输入。可以在命令行只指定用户名，程序运行后会提示输入口令。

虽然ydcmd主要供投资者使用，但是部分功能也可以被有需要的管理员使用，可被管理员使用的功能请参考下面各个功能的详细说明。

12.1.1 导出投资者数据工具

导出投资者的所有业务数据到多个文件中，输出的内容是收到notifyCaughtUp时的数据快照。管理员可以使用，管理员使用时导出该柜台上所有投资者的所有业务数据。输出文件清单请参考导出投资者数据。

运行程序时需要指定导出目录，若不指定则导出到当前目录。

12.1.2 修改口令工具

修改投资者口令。管理员可以使用，管理员使用时修改管理员自身口令。

新口令可以在命令行指定。如果没有在命令行中指定新口令，程序运行后会提示输入两次新口令。

12.1.3 全撤报单工具

撤销投资者处于挂单状态的全部普通报单，具体包括：

- 挂单状态的普通报单，包括触发单
- 挂单状态的报价派生报单，且该交易所允许撤报价派生报单

请注意，非普通报单（YDOrderFlag不为0）不在撤销范围内，例如行权、放弃自动行权等。

如果参数中指定了once，程序将按照收到notifyCaughtUp时的普通挂单列表逐一发出撤单指令后就结束运行，不会等待结果，也不会再次发出撤单指令，命令行上会输出实际发出的撤单数量。若挂单数量巨大，可能会触发柜台流控或者交易所流控，因此一次运行可能无法确保全部撤单，此时可多次运行或使用交互撤单模式。

在使用once模式时，程序的返回值定义如下：

- 0：没有可撤的报单
- 1：其他各类错误
- 2：有可撤的报单，且api发出了所有撤单指令
- 3：有可撤的报单，在api发出撤单指令时发生了发送失败

如果参数中未指定once，程序将交互式运行撤单逻辑，命令行上会持续刷新如下信息：

```
XXX order(s) can be canceled, press return to start
```

投资者每按回车一次，就会对当前可撤销报单发出撤单指令，然后在下一行继续刷新可撤单数量。当可撤报单数为0时，程序自动结束。

12.1.4 全撤报价工具

撤销投资者至少存在一侧的报单还处于挂单状态的全部报价。

如果参数中指定了once，程序将按照收到notifyCaughtUp时的可撤报价列表逐一发出撤报价指令后就结束运行，不会等待结果，也不会再次发出撤报价指令，命令行上会输出实际发出的撤报价数量。若可撤报价数量巨大，可能会触发柜台流控或者交易所流控，因此一次运行可能无法确保全部撤报价，此时可多次运行或使用交互撤报价模式。

在使用once模式时，程序的返回值定义如下：

- 0：没有可撤的报价
- 1：其他各类错误
- 2：有可撤的报价，且api发出了所有撤报价指令
- 3：有可撤的报价，在api发出撤报价指令时发生了发送失败

如果参数中未指定once，程序将交互式运行撤报价逻辑，命令行上会持续刷新如下信息：

```
XXX quote(s) can be canceled, press return to start
```

投资者每按回车一次，就会对当前可撤销报价发出撤报价指令，然后在下一行继续刷新可撤报价数量。当可撤报价单数为0时，程序自动结束。

12.2 日初数据核对工具

日初数据核对工具ydInitVerify可以帮助经纪商管理员和投资者核对某套易达柜台的日初持仓与主席柜台的日初持仓的一致性。

本工具的检查范围如下所示：

- 业务范围。本工具支持检查沪深交易所现货、股票期权，期货交易所的期货和期权等易达柜台支持的所有类型日初持仓。本工具的核对对象是日初持仓，并不会检查当前持仓和资金。各类型持仓支持的系统数据源如下：
  - 沪深交易所现货：只支持使用文件数据源，文件数据源的格式参见文件比对格式
  - 沪深交易所股票期权：只支持通过CTP股票期权系统交易接口核对
  - 期货交易所的期货和期权：只支持通过CTP期货交易系统接口核对
- 数据源范围。在配置文件中指定的数据源，可以是其中一个或者是多个数据源的任意组合。若配置了多个数据源，数据源的读取顺序为文件、CTP期货、CTP股票期权。如果不同数据源对同一个账户合约提供了数据，本工具会告警并按顺序使用靠前的数据源中的数据继续运行
- 资金账号范围。管理员登录时核对易达柜台上所有资金账号的日初持仓，投资者登录时核对该投资者的资金账号的日初持仓
- 交易所范围。默认检查易达柜台上存在的所有交易所，可以通过配置文件中的verify.Exchange将交易所限定为默认交易所的子集

只要所需数据源可达，本工具可以在任意时间使用，即便在开盘后也可正常检查日初持仓。使用前请按照日初数据核对工具配置完成工具配置，然后通过执行以下命令启动本工具。运行过程信息会输出到标准输出，如果运行过程中有任何错误，会在标准输出上显示错误信息，并返回非0退出值。

```
./ydInitVerify
```

12.2.1 日初数据核对工具配置

本工具支持Win64和Linux64两个平台的版本，请先根据使用的操作系统解压ydClient压缩中的ydInitVerify的对应版本目录至指定位置，然后依次完成下列配置步骤：

- 配置yd.ini中要连接的易达柜台的信息，配置信息说明请参见配置文件。
- 配置yd.ini中供ydInitVerify使用的核对信息，配置信息说明参见下表

参数分类	参数	说明
核对配置	verify.Exchange	需要核对的交易所。可以配置多个。如果为空，表示核对所有易达系统上的交易所
易达客户端配置	yd.Username	登录易达的管理员账号或者资金账号。必填
	yd.Password	登录易达的口令。若为空，会在命令行上提示输入
	yd.Timeout	查询易达日初数据的超时时间，单位为秒。缺省30秒
CTP期货数据源	ctp.TradeURL	CTP期货数据源的地址。若为空，表示不核对CTP期货日初数据，此时CTP期货数据源配置均无效
	ctp.BrokerID	CTP期货数据源的会员号。必填
	ctp.Username	登录CTP期货数据源的管理员账号或者资金账号。必填
	ctp.Password	登录CTP期货数据源的口令。若为空，会在命令行上提示输入
	ctp.AppID	登录CTP期货数据源的AppID。若为空，表示登录CTP后不认证AppID
	ctp.AuthCode	登录CTP期货数据源的AuthCode。缺省为空
	ctp.UserProductInfo	登录CTP期货数据源的用户产品信息。缺省为空
	ctp.Timeout	查询CTP期货日初数据的超时时间，单位为秒，缺省30秒
CTP股票期权数据源	ctpStock.TradeURL	CTP股票期权数据源的地址。若为空，表示不核对CTP股票期权日初数据，此时CTP股票期权数据源配置均无效
	ctpStock.BrokerID	CTP股票期权数据源的会员号。必填
	ctpStock.Username	登录CTP股票期权数据源的管理员账号或者资金账号。必填
	ctpStock.Password	登录CTP股票期权数据源的口令。若为空，会在命令行上提示输入
	ctpStock.AppID	登录CTP股票期权数据源的AppID。若为空，表示登录CTP后不认证AppID
	ctpStock.AuthCode	登录CTP股票期权数据源的AuthCode。缺省为空
	ctpStock.UserProductInfo	登录CTP股票期权数据源的用户产品信息。缺省为空
	ctpStock.Timeout	查询CTP股票期权日初数据的超时时间，单位为秒，缺省30秒
文件数据源	file.Name	文件名，如果有就核对，没有就表示不使用该数据源

12.2.2 文件比对格式

如果使用文件比对，则需将待比对的日初数据转换为以下格式，并设置文件比对配置参数后，才能开始比对。

字段	说明
AccountID	资金账号
InstrumentID	合约代码
ExchangeID	交易所代码
PositionType	持仓类型，可选字段 如果有本字段，正偶数表示多头，正奇数表示空头 如果没有本字段，正数持仓量表示多头，负数持仓量表示空头
Position	持仓量